TSF Project

Solution and Model report

GREAT LAKES INSTITUTE OF MANAGEMENT

Table of Contents

TIME SERIES Assignment

The attached data shows monthly demand of two different types of consumable items in a certain store from January 2002 to September 2017. The ultimate objective of this exercise is to predict sales for the period October 2017 to December 2018.

I. Read the data as time series objects in R. Plot the data. What are the major features you notice in the series? How do the two series differ?

II. Before a formal extraction of time series components is done, can you check for seasonal changes in the data for the two series separately? Particularly whether there are more variability in a season compared to the others, whether seasonal variations are changing across years etc. Compare the behaviour of the two series.

III. Decompose each series to extract trend and seasonality, if there are any. Which seasonality is more appropriate – additive or multiplicative? Explain the seasonal indices. In which month(s) do you see higher sales and which month(s) you see lower sales? Any difference in the nature of demand of the two items?

IV. Can you extract the residuals for the two decomposition exercises and check if they form a stationary series? Do a formal test for stationary writing down the null and alternative hypothesis. What is your conclusion in each case?

V. Before the final forecast is undertaken one would like to compare a few models. Use the last 21 months as hold-out sample fit a suitable exponential smoothing model to the rest of the data and calculate MAPE. What are the values of α, β and γ? What role do they play in the modeling? For the same hold-out period compare forecast by decomposition and compute MAPE. Which model gives smaller MAPE? Give a comparison for the two demands.

VI. Use the 'best' model obtained from above to forecast demand for the period Oct 2017 to December 2018 for both items. Provide forecasted values as well as their upper and lower confidence limits. If you are the store manager what decisions would you make after looking at the demand of the two items over years?

Response:

In the given dataset we have demand data for Item A and B for the period January 2002 to July 2017 (*as against Sep 2017 mentioned in the problem statement*). The data is continuous monthly data for the whole period without any breaks. This qualifies for a time series analysis on the demand for Item A & B, subject to other assumptions being valid.

The plan is to do the following:

- Convert to time series for the 2 types of data
- Perform Exploratory analysis of data by visual check of trend, seasonality
- Time series decomposition into trend, seasonality and residuals
- Check assumptions for stationary time series. In case of non-stationary, convert to stationary

- Create various models of Forecasting the time series value.
- Interpretation of result

Clean the environment and the memory

```
rm(list=ls())
gc()
```

```
##           used (Mb) gc trigger (Mb) max used (Mb)
## Ncells  480945  25.7     940480  50.3   750400 40.1
## Vcells  906393   7.0    1650153  12.6  1150485  8.8
```

Reading the dataset

Removing the first record from the excel

```
library(readxl)
#demand <- read_excel(file.choose(),skip = 1)
demand <-read_excel("~/Downloads/Demand.xlsx",skip=1)
```

```
## Warning in strptime(x, format, tz = tz): unknown timezone 'zone/tz/2018c.
## 1.0/zoneinfo/Asia/Kolkata'
```

```
str(demand)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':187 obs. of  4 variables:
##  $ Year   : num  2002 2002 2002 2002 2002 ...
##  $ Month  : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ Item A: num  1954 2302 3054 2414 2226 ...
##  $ Item B: num  2585 3368 3210 3111 3756 ...
```

Name the columns as per the series

```
names(demand)[3] <- c("ItemA")
names(demand)[4] <- c("ItemB")
str(demand)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':187 obs. of  4 variables:
##  $ Year : num  2002 2002 2002 2002 2002 ...
##  $ Month: num  1 2 3 4 5 6 7 8 9 10 ...
##  $ ItemA: num  1954 2302 3054 2414 2226 ...
##  $ ItemB: num  2585 3368 3210 3111 3756 ...
```

```
summary(demand)
```

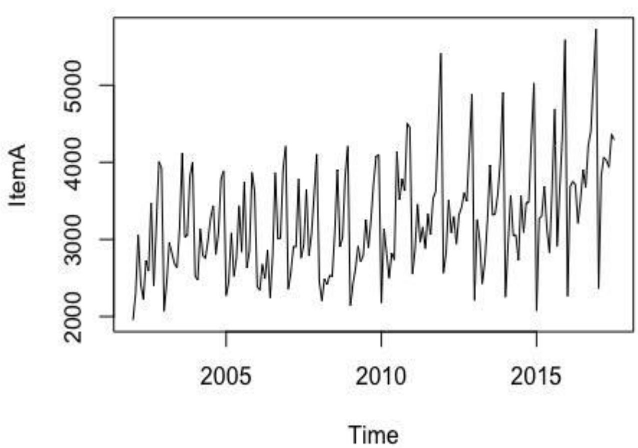
```
##       Year          Month            ItemA          ItemB
## Min.    :2002   Min.   : 1.000   Min.    :1954   Min.   :1153
##  1st Qu.:2005   1st Qu.: 3.000    1st Qu.:2748   1st Qu.:2362
## Median :2009   Median :  6.000   Median :3134   Median :2876
## Mean    :2009   Mean    : 6.406   Mean    :3263   Mean    :2962
```

```
##   3rd Qu.:2013   3rd Qu.: 9.000   3rd Qu.:3741   3rd Qu.:3468  
##   Max.   :2017   Max.   :12.000   Max.   :5725   Max.   :5618
```

Loading the Series onto a data frame and plot

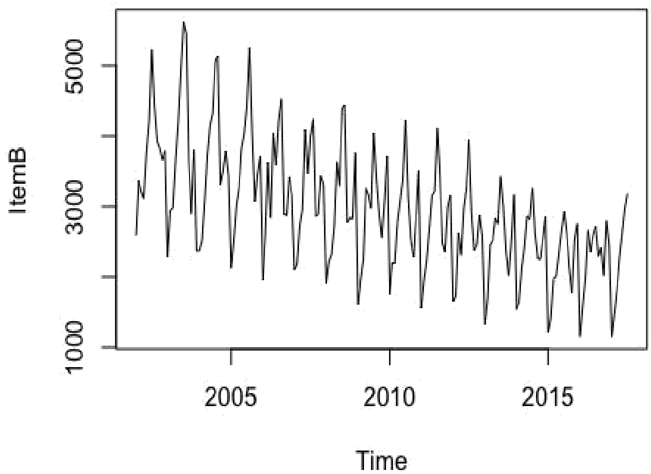
Plottig Time Series for Item A for Monthly data from year 2012 Jan to 2017 July

```
dem_ItA <- ts (demand[,3], start=c(2002,1), end=c(2017,7), frequency=12)
plot(dem_ItA)
```



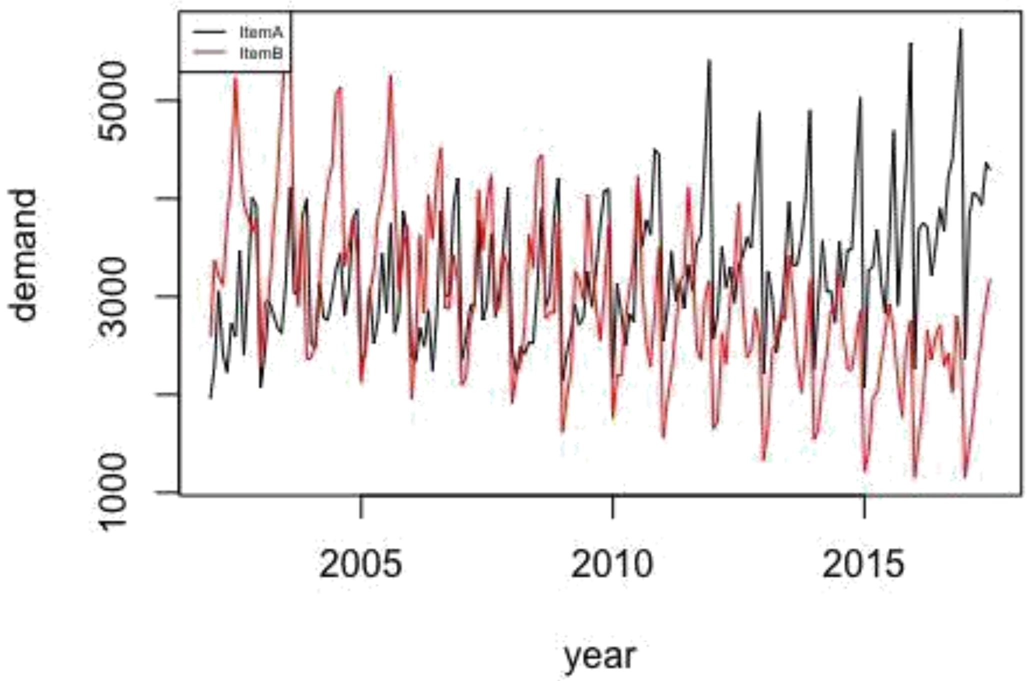
Plottig Time Series for Item B for Monthly data from year 2012 Jan to 2017 July

```
dem_ItB <- ts (demand[,4], start=c(2002,1), end=c(2017,7), frequency=12)
plot(dem_ItB)
```



Plotting the Time Series across Item A and Item B

```
ts.plot(dem_ItA, dem_ItB, gpars = list(col = c("black", "red")),xlab="year",
ylab="demand")
legend("topleft", colnames(demand[3:4]), col=1:ncol(demand), lty=1.9, cex=.45
)
```



From the above plots, we can see Item A has an increasing demand, whereas Item B has fall in demand. Also, from big picture we can see some seasonality and trend in demands. Both Item A and B doesn't seem to have cyclic in nature. Item A the variation increases with time and Item B decreases.

Decomposition of time series

A time series decomposition is procedure which transform a time series into multiple different time series. The original time series is often computed (decompose) into 3 sub-time series:

Seasonal: patterns that repeat with fixed period of time. Trend: the underlying trend of the metrics. Random: (also call "noise", "Irregular" or "Remainder") Is the residuals of the time series after allocation into the seasonal and trends time series. Other than above three component there is Cyclic component which occurs after long period of time

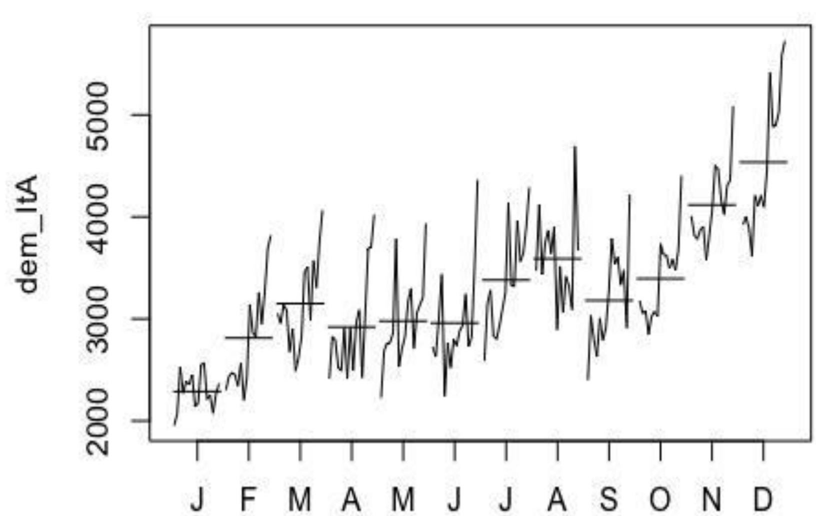
Additive or multiplicative decomposition?

To get a successful decomposition, it is important to choose between the additive or multiplicative model. To choose the right model we need to look at the time series.

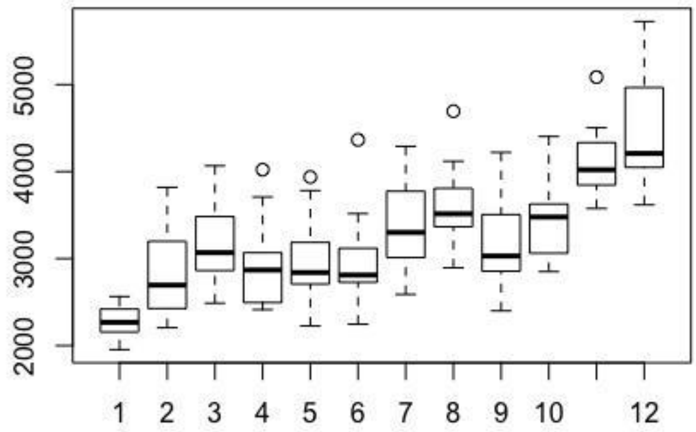
a. The additive model is useful when the seasonal variation is relatively constant over time.
b. The multiplicative model is useful when the seasonal variation increases over time.

Item A

```
monthplot(dem_ItA)
```



```r
boxplot(dem_ItA ~cycle(dem_ItA))
```

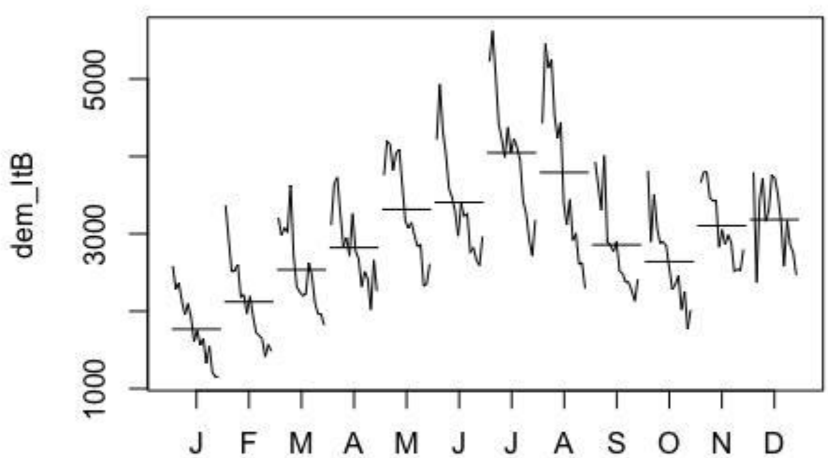


I

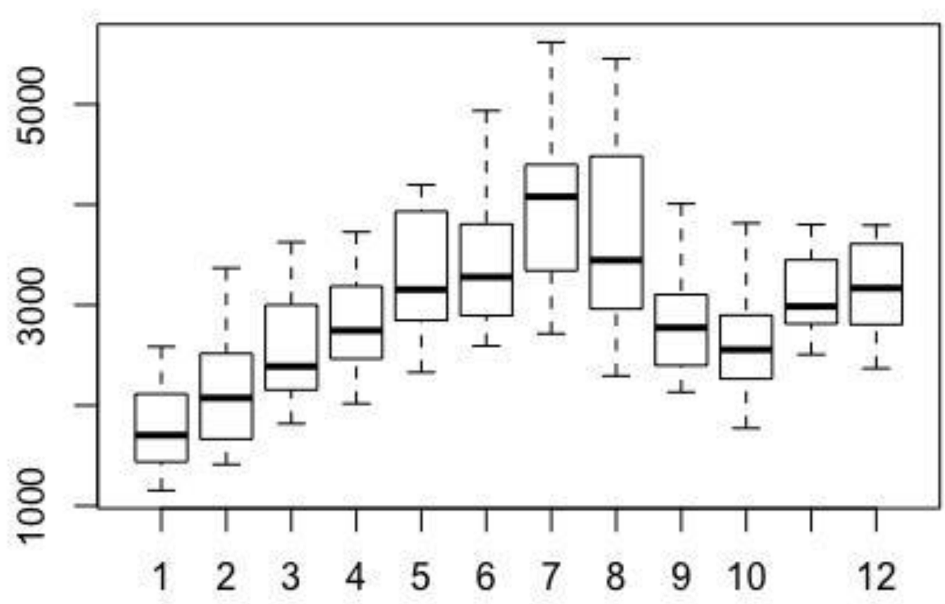
tem A has maximum variation in May, Sep, Oct, Dec. Also, can few outlier in the month of Apr, May, June, Aug, Nov.

Item B

```r
monthplot(dem_ItB)
```



```
boxplot(dem_ItB ~cycle(dem_ItB))
```



Item B has maximum variations in Jun, Jul, Aug. There seems to no outliers.

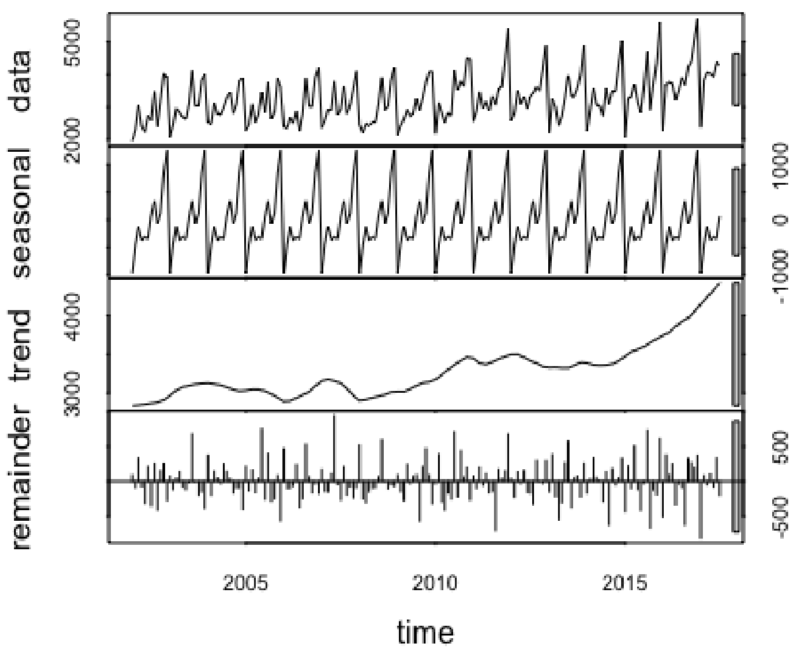
The seasonal variation looked to be about the same magnitude across time, so an additive decomposition might give good results.

STL

STL is a very versatile and robust method for decomposing time series. STL is an acronym for “Seasonal and Trend decomposition using Loess”. It does an additive decomposition and the four graphs are the original data, seasonal component, trend component and the remainder.

ITEM A

```
ItA_Sea <- stl(dem_ItA[,1], s.window="p") #constant
seasonality plot(ItA_Sea)
```




```
ItA_Sea

##  Call:
##  stl(x = dem_ItA[, 1], s.window = "p")
##
## Components
##             seasonal    trend    remainder
## Jan 2002 -970.47187 2838.594   85.8774947
## Feb 2002 -454.22689 2841.712  -85.4854465
## Mar 2002 -124.79419 2844.830  333.9638914
## Apr 2002 -364.03321 2850.118  -72.0843134
## May 2002 -314.83443 2855.405 -314.5703082
## Jun 2002 -343.58304 2861.823  206.7602840
## Jul 2002   68.54396 2868.241 -347.7847307
## Aug 2002  342.76630 2874.802  252.4314442
## Sep 2002  -73.74798 2881.364 -407.6157649
## Oct 2002  131.64891 2896.590  151.7607193
## Nov 2002  845.91259 2911.817  251.2704127
## Dec 2002 1256.81992 2940.849 -273.6689000
## Jan 2003  -970.47187 2969.881   72.5909150
## Feb  2003 -454.22689 3003.352 -115.1249435
## Mar  2003 -124.79419 3036.823   43.9714772
## Apr 2003 -364.03321 3057.338  134.6955475
## May 2003 -314.83443 3077.853  -76.0181722
## Jun 2003 -343.58304 3087.191 -114.6081609
## Jul 2003    68.54396 3096.530  -15.0737565
## Aug  2003  342.76630 3104.797  671.4370969
## Sep  2003  -73.74798  3113.063   -9.3154337
## Oct 2003  131.64891 3118.789 -195.4380889
## Nov 2003  845.91259 3124.515 -149.4275348
## Dec 2003 1256.81992 3126.694 -382.5135585
## Jan 2004 -970.47187 3128.872  370.5995456
## Feb 2004 -454.22689 3122.892 -196.6654794
## Mar 2004 -124.79419 3116.912  141.8817748
## Apr  2004 -364.03321  3109.198   43.8349552
## May  2004 -314.83443  3101.484  -28.6496543
```

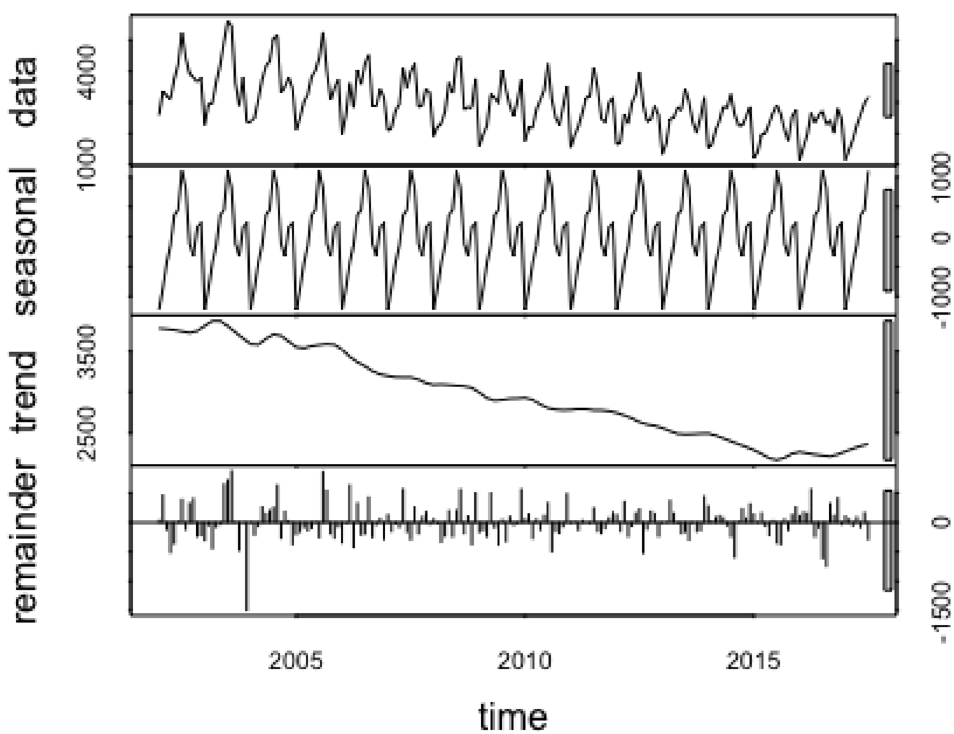
```
## Jun 2004 -343.58304 3088.648  247.9347027
## Jul 2004   68.54396 3075.813  137.6434526
## Aug 2004  342.76630 3059.784   34.4500405
## Sep 2004  -73.74798 3043.755 -166.0067556
## Oct 2004  131.64891 3034.905  -90.5534246
## Nov 2004  845.91259 3026.054  -89.9668843
## Dec 2004 1256.81992 3029.246 -397.0656630
## Jan 2005 -970.47187 3032.437  209.0346861
## Feb 2005 -454.22689 3039.959 -133.7323126
## Mar 2005 -124.79419 3047.481  161.3129679
## Apr 2005 -364.03321 3047.059 -161.0258089
## May 2005 -314.83443 3046.637   37.1976242
## Jun 2005 -343.58304 3036.439  745.1440056
## Jul 2005   68.54396 3026.241 -255.7852200
## Aug 2005  342.76630 3008.037  395.1969295
## Sep 2005  -73.74798 2989.832 -284.0843049
## Oct 2005  131.64891 2966.165 -246.8140342
## Nov 2005  845.91259 2942.498   82.5894458
## Dec 2005 1256.81992 2918.695 -557.5150968
## Jan 2006 -970.47187 2894.892  464.5794884
## Feb 2006 -454.22689 2897.339  -99.1125436
## Mar 2006 -124.79419 2899.786  -96.9922965
## Apr 2006 -364.03321 2918.733  -62.6996684
## May 2006 -314.83443 2937.679  235.1551698
## Jun 2006 -343.58304 2957.769 -368.1864319
## Jul 2006   68.54396 2977.860 -246.4036406
## Aug 2006  342.76630 2997.834  528.4001189
## Sep 2006  -73.74798 3017.807   62.9404943
## Oct 2006  131.64891 3054.752 -163.4013810
## Nov 2006  845.91259 3091.697  -30.6100471
## Dec 2006 1256.81992 3124.010 -171.8295606
## Jan 2007 -970.47187 3156.322  167.1500535
## Feb 2007 -454.22689 3166.228 -142.0007290
## Mar 2007 -124.79419 3176.133 -148.3392323
## Apr 2007 -364.03321 3166.731  107.3025511
## May 2007 -314.83443 3157.328  939.5065446
## Jun 2007 -343.58304 3142.154  -39.5704640
## Jul 2007   68.54396 3126.979 -264.5230796
## Aug 2007  342.76630 3093.841  204.3926298
## Sep 2007  -73.74798 3060.703 -192.9550449
## Oct 2007  131.64891 3013.480  -75.1292433
## Nov 2007  845.91259 2966.258 -236.1702325
## Dec 2007 1256.81992 2935.676  -86.4962591
## Jan 2008 -970.47187 2905.095  517.3768420
## Feb 2008 -454.22689 2910.809 -250.5824819
## Mar 2008 -124.79419 2916.524 -303.7295265
## Apr 2008 -364.03321 2928.387 -148.3537341
## May 2008 -314.83443 2940.250  -91.4157316
## Jun 2008 -343.58304 2950.005  -85.4223439
## Jul 2008   68.54396 2959.761   64.6954368
## Aug 2008  342.76630 2970.720  589.5136280
## Sep 2008  -73.74798 2981.680   -0.9315648
## Oct 2008  131.64891 2997.095 -103.7437291
## Nov 2008  845.91259 3012.510  -46.4226841
## Dec 2008 1256.81992 3015.926  -63.7454912
## Jan 2009 -970.47187 3019.341   89.1308295
## Feb 2009 -454.22689 3019.088 -145.8611006
## Mar 2009 -124.79419 3018.835 -272.0407516
## Apr 2009 -364.03321 3035.684  240.3492488
## May 2009 -314.83443 3052.533  -29.6985406
## Jun 2009 -343.58304 3074.361   67.2219438
## Jul 2009   68.54396 3096.189   89.2668213
## Aug 2009  342.76630 3113.468 -561.2347419
## Sep 2009  -73.74798 3130.748  206.0003110
## Oct 2009  131.64891 3137.518  466.8329400
## Nov 2009  845.91259 3144.289   86.7987783
## Dec 2009 1256.81992 3159.694 -319.5136986
## Jan 2010 -970.47187 3175.099  -29.6270477
## Feb 2010 -454.22689 3206.525  385.7013958
```

```
## Mar 2010 -124.79419 3237.952 -290.1578815
## Apr 2010 -364.03321 3270.367 -408.3342287
## May 2010 -314.83443 3302.783 -165.9483659
## Jun 2010 -343.58304 3333.471 -251.8883314
## Jul 2010   68.54396 3364.160  704.2960960
## Aug 2010  342.76630 3392.636 -220.4025750
## Sep 2010  -73.74798 3421.113  437.6353699
## Oct 2010  131.64891 3443.006   57.3450184
## Nov 2010  845.91259 3464.900  193.1878762
## Dec 2010 1256.81992 3456.400 -262.2202249
## Jan 2011 -970.47187 3447.901   72.5708018
## Feb 2011 -454.22689 3418.038  -96.8109075
## Mar 2011 -124.79419 3388.175  194.6196624
## Apr 2011 -364.03321 3378.961  -53.9280669
## May 2011 -314.83443 3369.748  108.0864138
## Jun 2011 -343.58304 3386.786 -163.2027125
## Jul 2011   68.54396 3403.823 -141.3674457
## Aug 2011  342.76630 3421.607 -702.3734241
## Sep 2011  -73.74798 3439.391  168.3572135
## Oct 2011  131.64891 3455.489   34.8623589
## Nov 2011  845.91259 3471.587  146.5007135
## Dec 2011 1256.81992 3485.000  669.1799162
## Jan 2012 -970.47187 3498.414   36.0582467
## Feb 2012 -454.22689 3499.196 -224.9688933
## Mar 2012 -124.79419 3499.978  132.8162458
## Apr 2012 -364.03321 3480.838  -28.8052222
## May 2012 -314.83443 3461.699  152.1355198
## Jun 2012 -343.58304 3440.370 -157.7869009
## Jul 2012   68.54396 3419.041 -167.5849286
## Aug 2012  342.76630 3402.872 -327.6382310
## Sep 2012  -73.74798 3386.703  291.0450826
## Oct 2012  131.64891 3364.943   -1.5918378
## Nov 2012  845.91259 3343.183  -26.0955489
## Dec 2012 1256.81992 3336.932  288.2483373
## Jan 2013 -970.47187 3330.681 -149.2086487
## Feb 2013 -454.22689 3332.265  381.9617543
## Mar 2013 -124.79419 3333.850 -217.0555635
## Apr 2013 -364.03321 3329.875 -540.8417226
## May 2013 -314.83443 3325.900 -304.0656715
## Jun 2013 -343.58304 3326.168  261.4152929
## Jul 2013   68.54396 3326.435  570.0206504
## Aug 2013  342.76630 3342.291 -370.0573352
## Sep 2013  -73.74798 3358.147   48.6012952
## Oct 2013  131.64891 3377.263   74.0876494
## Nov 2013  845.91259 3396.380 -221.2927871
## Dec 2013 1256.81992 3392.910  254.2703403
## Jan 2014 -970.47187 3389.439 -166.9674046
## Feb 2014  -454.22689  3375.881   30.3455424
## Mar 2014  -124.79419  3362.323  335.4707685
## Apr  2014 -364.03321  3359.708   52.3247415
## May  2014 -314.83443  3357.094   16.7409246
## Jun 2014 -343.58304 3358.790 -284.2070042
## Jul 2014   68.54396 3360.487  133.9694600
## Aug 2014  342.76630 3368.331 -619.0974322
## Sep 2014  -73.74798 3376.176  175.5722915
## Oct 2014  131.64891 3395.621  -49.2697962
## Nov 2014  845.91259 3415.066   47.0213253
## Dec 2014 1256.81992 3445.268  326.9116229
## Jan 2015 -970.47187 3475.471 -429.9989519
## Feb 2015 -454.22689 3503.116  215.1105138
## Mar 2015 -124.79419 3530.762  -97.9677414
## Apr 2015 -364.03321 3547.556  504.4773960
## May 2015 -314.83443 3564.350 -113.5152566
## Jun 2015 -343.58304 3582.039 -414.4556491
## Jul 2015   68.54396 3599.728  -24.2716486
## Aug 2015  342.76630 3625.370  725.8635970
## Sep 2015  -73.74798 3651.013 -663.2645415
## Oct 2015  131.64891 3675.729 -121.3777786
## Nov 2015  845.91259 3700.445 -188.3578064
```

```
## Dec 2015 1256.81992 3722.346  607.8341879
## Jan 2016 -970.47187 3744.247 -508.7746901
## Feb 2016 -454.22689 3764.933  374.2938336
## Mar 2016 -124.79419 3785.620   93.1746364
## Apr 2016 -364.03321 3823.905  248.1284001
## May 2016 -314.83443 3862.190 -337.3556261
## Jun 2016 -343.58304 3891.765  -31.1819273
## Jul 2016   68.54396 3921.340  -84.8838355
## Aug 2016  342.76630 3945.568 -618.3341758
## Sep 2016  -73.74798 3969.796  324.9520999
## Oct 2016  131.64891 4009.832  262.5194295
## Nov 2016  845.91259 4049.867  190.2199684
## Dec 2016 1256.81992 4098.522  369.6584691
## Jan 2017 -970.47187 4147.176 -809.7039024
## Feb 2017 -454.22689 4192.984   80.2429755
## Mar 2017 -124.79419 4238.792  -46.9978675
## Apr 2017 -364.03321 4283.743  102.2902147
## May 2017 -314.83443 4328.694  -76.8594931
## Jun 2017 -343.58304 4373.977  334.6055674
## Jul 2017   68.54396 4419.261 -197.8049790
```

ITEM B

```
ItB_Sea<-stl(dem_ItB[,1], s.window="p") #constant seasonality
plot(ItB_Sea)
```




```
ItB_Sea

##  Call:
##  stl(x = dem_ItB[, 1], s.window = "p")
```

```
##
## Components
##             seasonal    trend     remainder
## Jan 2002 -1222.0193 3777.014    30.0049071
## Feb 2002  -860.5255 3773.067   455.4584776
## Mar 2002  -438.6564 3769.120  -120.4632005
## Apr 2002  -145.5874 3763.765  -507.1773823
## May 2002   354.4193 3758.410  -356.8292167
## Jun 2002   451.4649 3749.684    14.8509274
## Jul 2002  1098.3861 3740.958   385.6555393
## Aug 2002   813.8549 3732.790  -120.6447214
## Sep 2002  -113.5380 3724.621   320.9166729
## Oct 2002  -322.9173 3735.440   403.4769361
## Nov 2002   148.5034 3746.259  -233.7626867
## Dec 2002   236.6154 3777.311  -218.9267939
## Jan 2003 -1222.0193 3808.364  -301.3442675
## Feb 2003  -860.5255 3838.578   -44.0525816
## Mar 2003  -438.6564 3868.793  -445.1361442
## Apr 2003  -145.5874 3868.446   -76.8583167
## May 2003   354.4193 3868.099   -24.5181418
## Jun 2003   451.4649 3836.751   646.7841038
## Jul 2003  1098.3861 3805.403   714.2108172
## Aug 2003   813.8549 3767.750   872.3955424
## Sep 2003  -113.5380 3730.096     7.4419225
## Oct 2003  -322.9173 3693.110  -472.1928573
## Nov 2003   148.5034 3656.124    -2.6275230
## Dec 2003   236.6154 3621.627 -1489.2423067
## Jan 2004 -1222.0193 3587.130     3.8895432
## Feb 2004  -860.5255 3583.290  -211.7649189
## Mar 2004  -438.6564 3579.451   -61.7946295
## Apr 2004  -145.5874 3614.744   258.8429414
## May 2004   354.4193 3650.038   146.5428596
## Jun 2004   451.4649 3677.542   196.9926637
## Jul 2004  1098.3861 3705.047   250.5669357
## Aug 2004   813.8549 3695.809   628.3356572
## Sep 2004  -113.5380 3686.572  -263.0339664
## Oct 2004  -322.9173 3651.969   178.9481439
## Nov 2004   148.5034 3617.366    24.1303682
## Dec 2004   236.6154 3582.613  -373.2287132
## Jan 2005 -1222.0193 3547.860  -198.8411610
## Feb 2005  -860.5255 3542.228  -158.7025912
## Mar 2005  -438.6564 3536.596   -80.9392700
## Apr 2005  -145.5874 3548.537  -137.9492169
## May 2005   354.4193 3560.478   -92.8968165
## Jun 2005   451.4649 3568.072     7.4632162
## Jul 2005  1098.3861 3575.666  -254.0522834
## Aug 2005   813.8549 3581.249   859.8957788
## Sep 2005  -113.5380 3586.833   535.7054958
## Oct 2005  -322.9173 3582.274  -185.3571167
## Nov 2005   148.5034 3577.716  -261.2196153
## Dec 2005   236.6154 3549.095   -67.7106744
## Jan 2006 -1222.0193 3520.474  -344.4550999
## Feb 2006  -860.5255 3476.768   -12.2429342
## Mar 2006  -438.6564 3433.062   631.5939829
## Apr 2006  -145.5874 3399.776  -418.1883478
## May 2006   354.4193 3366.489   321.0916690
## Jun 2006   451.4649 3341.434  -208.8987092
## Jul 2006  1098.3861 3316.379  -189.7646195
## Aug 2006   813.8549 3285.884   423.2607350
## Sep 2006  -113.5380 3255.390  -249.8522556
## Oct 2006  -322.9173 3235.904   -36.9862712
## Nov 2006   148.5034 3216.417    55.0798272
## Dec 2006   236.6154 3207.012  -284.6273818
## Jan 2007 -1222.0193 3197.607   125.4120429
## Feb 2007  -860.5255 3189.779  -148.2533388
## Mar 2007  -438.6564 3181.950   -19.2939689
## Apr 2007  -145.5874 3180.470   -80.8825977
## May 2007   354.4193 3178.990   558.5911210
## Jun 2007   451.4649 3179.044  -160.5085347
```

```
## Jul 2007  1098.3861 3179.098  -287.4837226
## Aug 2007   813.8549 3166.201   258.9437112
## Sep 2007  -113.5380 3153.305  -184.7672001
## Oct 2007  -322.9173 3131.548    88.3691999
## Nov 2007   148.5034 3109.791   174.7057139
## Dec 2007   236.6154 3098.356   -27.9718498
## Jan 2008 -1222.0193 3086.922    49.0972201
## Feb 2008  -860.5255 3089.208   -14.6824028
## Mar 2008  -438.6564 3091.494  -332.8372742
## Apr 2008  -145.5874 3088.438  -228.8507627
## May 2008   354.4193 3085.383   193.1980963
## Jun 2008   451.4649 3081.804  -238.2689104
## Jul 2008  1098.3861 3078.225   200.3885508
## Aug 2008   813.8549 3074.835   553.3100076
## Sep 2008  -113.5380 3071.445  -183.9068806
## Oct 2008  -322.9173 3061.875   101.0422257
## Nov 2008   148.5034 3052.305  -372.8085539
## Dec 2008   236.6154 3022.876   498.5086522
## Jan 2009 -1222.0193 2993.447  -161.4275081
## Feb 2009  -860.5255 2959.706  -131.1802622
## Mar 2009  -438.6564 2925.965  -239.3082648
## Apr 2009  -145.5874 2910.884   496.7032162
## May 2009   354.4193 2895.804   -86.2229554
## Jun 2009   451.4649 2900.145  -379.6095808
## Jul 2009  1098.3861 2904.486    38.1282617
## Aug 2009   813.8549 2911.501  -323.3554940
## Sep 2009  -113.5380 2918.516    93.0224053
## Oct 2009  -322.9173 2919.904   -41.9870368
## Nov 2009   148.5034 2921.293   -13.7963649
## Dec 2009   236.6154 2924.851   555.5336462
## Jan 2010 -1222.0193 2928.409    48.6102910
## Feb 2010  -860.5255 2916.592   136.9330544
## Mar 2010  -438.6564 2904.776  -268.1194306
## Apr 2010  -145.5874 2876.726    45.8612011
## May 2010   354.4193 2848.677  -127.0958199
## Jun 2010   451.4649 2826.763   110.7717431
## Jul 2010  1098.3861 2804.850   327.7637739
## Aug 2010   813.8549 2795.724  -491.5793469
## Sep 2010  -113.5380 2786.599  -149.0608128
## Oct 2010  -322.9173 2783.078  -180.1612184
## Nov 2010   148.5034 2779.558   -66.0615100
## Dec 2010   236.6154 2781.363   484.0216829
## Jan 2011 -1222.0193 2783.168    -3.1484905
## Feb 2011  -860.5255 2788.284    12.2410865
## Mar 2011  -438.6564 2793.401  -128.7445851
## Apr 2011  -145.5874 2792.256    29.3318600
## May 2011   354.4193 2791.110    -0.5293475
## Jun 2011   451.4649 2785.008   -12.4732759
## Jul 2011  1098.3861 2778.907   239.7072636
## Aug 2011   813.8549 2778.112  -145.9668971
## Sep 2011  -113.5380 2777.317  -181.7794028
## Oct 2011  -322.9173 2772.347  -100.4292510
## Nov 2011   148.5034 2767.376    70.1210148
## Dec 2011   236.6154 2757.585   168.7993470
## Jan 2012 -1222.0193 2747.795   125.2243128
## Feb 2012  -860.5255 2732.067  -146.5413859
## Mar 2012  -438.6564 2716.339   344.3176670
## Apr 2012  -145.5874 2698.664  -237.0769482
## May 2012   354.4193 2680.990   -59.4092160
## Jun 2012   451.4649 2657.505   154.0302482
## Jul 2012  1098.3861 2634.020   218.5941802
## Aug 2012   813.8549 2619.763  -516.6183239
## Sep 2012  -113.5380 2605.507  -111.9691730
## Oct 2012  -322.9173 2597.660   183.2575145
## Nov 2012   148.5034 2589.812   144.6843161
## Dec 2012   236.6154 2576.275  -233.8900447
## Jan 2013 -1222.0193 2562.737   -10.7177719
## Feb 2013  -860.5255 2543.981     2.5440417
## Mar 2013  -438.6564 2525.226   370.4306067
```

```
## Apr 2013   -145.5874 2508.632   150.9552871
## May 2013    354.4193 2492.038   -12.4576850
## Jun 2013    451.4649 2488.810  -183.2745505
## Jul 2013   1098.3861 2485.581  -158.9669480
## Aug 2013    813.8549 2486.857  -294.7116972
## Sep 2013   -113.5380 2488.133    -5.5947915
## Oct 2013   -322.9173 2490.733  -150.8157453
## Nov 2013    148.5034 2493.333  -134.8365850
## Dec 2013    236.6154 2494.320   437.0646190
## Jan 2014 -1222.0193 2495.307   271.7124565
## Feb 2014   -860.5255 2483.294    20.2310288
## Mar 2014   -438.6564 2471.282    79.3743525
## Apr 2014   -145.5874 2453.692   106.8949770
## May 2014    354.4193 2436.103    71.4779490
## Jun 2014    451.4649 2418.038   -47.5026431
## Jul 2014   1098.3861 2399.973  -238.3587673
## Aug 2014    813.8549 2382.483  -590.3374754
## Sep 2014   -113.5380 2364.993    12.5454713
## Oct 2014   -322.9173 2347.987   224.9303472
## Nov 2014    148.5034 2330.981    65.5153372
## Dec 2014    236.6154 2312.344   307.0401754
## Jan 2015 -1222.0193 2293.708   136.3116472
## Feb 2015   -860.5255 2270.194     2.3319671
## Mar 2015   -438.6564 2246.679   155.9770385
## Apr 2015   -145.5874 2219.900   -56.3121661
## May 2015    354.4193 2193.120  -218.5390233
## Jun 2015    451.4649 2182.427    26.1085716
## Jul 2015   1098.3861 2171.733  -347.1193656
## Aug 2015    813.8549 2184.415  -372.2696947
## Sep 2015   -113.5380 2197.096    48.4416312
## Oct 2015  -322.9173 2222.920  -128.0029504
## Nov 2015   148.5034 2248.744   128.7525820
## Dec 2015   236.6154 2259.234   259.1502499
## Jan 2016 -1222.0193 2269.725   106.2945515
## Feb 2016  -860.5255 2261.681   166.8443606
## Mar 2016  -438.6564 2253.637   150.0189212
## Apr 2016  -145.5874 2245.677   558.9103102
## May 2016    354.4193 2237.717 -238.1359533
## Jun 2016    451.4649 2231.775   -91.2403271
## Jul  2016  1098.3861 2225.834 -610.2202330
## Aug  2016   813.8549 2220.731 -740.5854021
## Sep 2016  -113.5380 2215.627   313.9110837
## Oct 2016  -322.9173 2227.644   111.2731544
## Nov 2016   148.5034 2239.661   410.8353392
## Dec 2016    236.6154 2257.646  -27.2612198
## Jan 2017 -1222.0193 2275.630    99.3888549
## Feb 2017   -860.5255 2293.124   49.4013989
## Mar 2017   -438.6564  2310.618  -53.9613056
## Apr 2017   -145.5874  2326.261   81.3258918
## May 2017    354.4193  2341.905  -84.3245635
## Jun 2017   451.4649 2354.830   160.7048896
## Jul 2017  1098.3861 2367.755  -287.1411894
```

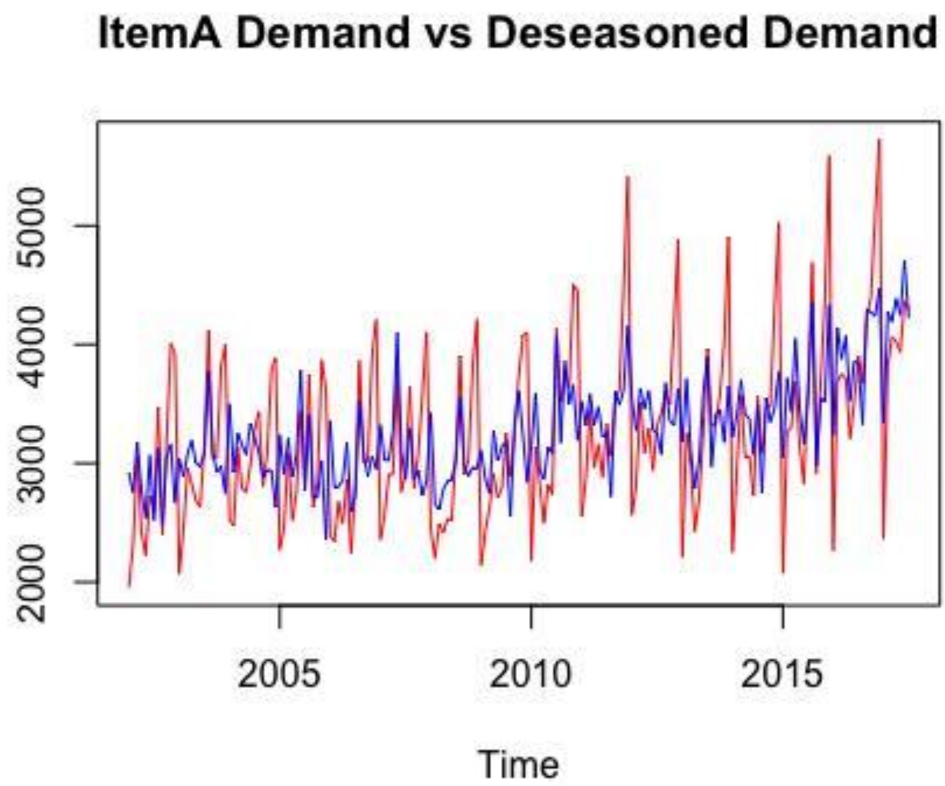
From the above decomposed details, we can see that there is continous incease in demand for Item A, but on contraty similar drop pattern observed for Item B.

Decompose the time series and plot the deseasoned series

If the focus is on figuring out whether the general trend of demand is up, we deseasonalize, and possibly forget about the seasonal component. However, if you need to forecast the demand in next mnoth, then you need take into account both the secular trend and seasonality.

Item A

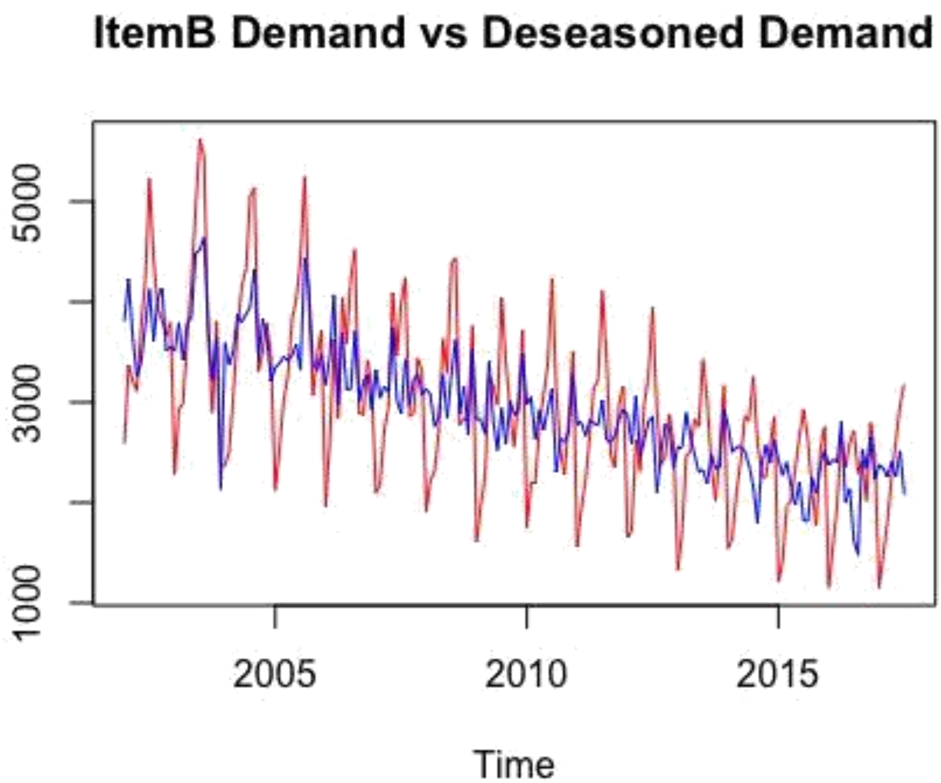
```
series_names <- c('Deseasoned', 'Actual')
Deseason_ItA <- (ItA_Sea$time.series[,2]+ItA_Sea$time.series[,3])
ts.plot(dem_ItA, Deseason_ItA, col=c("red", "blue"), main="ItemA Demand vs De
seasoned Demand")
```



The above plot show demand in Red and de-seasoned demand in Blue, we can see that there is increasing trend of demand. The residual component is also part of analysis.

Item B

```
Deseason_ItB <- (ItB_Sea$time.series[,2]+ItB_Sea$time.series[,3])
ts.plot(dem_ItB, Deseason_ItB, col=c("red", "blue"), main="ItemB Demand vs De
seasoned Demand")
```



The above plot show demand in Red and de-seasoned demand in Blue, we can see that there is decreasing trend of demand.

Divide data into test and train

```
DataATrain <- window(dem_ItA, start=c(2002,1), end=c(2015,12), frequency=12)
DataATest <- window(dem_ItA, start=c(2016,1), frequency=12)

DataBTrain <- window(dem_ItB, start=c(2002,1), end=c(2015,12), frequency=12)
DataBTest <- window(dem_ItB, start=c(2016,1), frequency=12)
```

Convert into seasonal, trend and irregular component suing STL

```
ItmATrn <- stl(DataATrain[,1], s.window="p")
ItmBTrn <- stl(DataBTrain[,1], s.window="p")
```

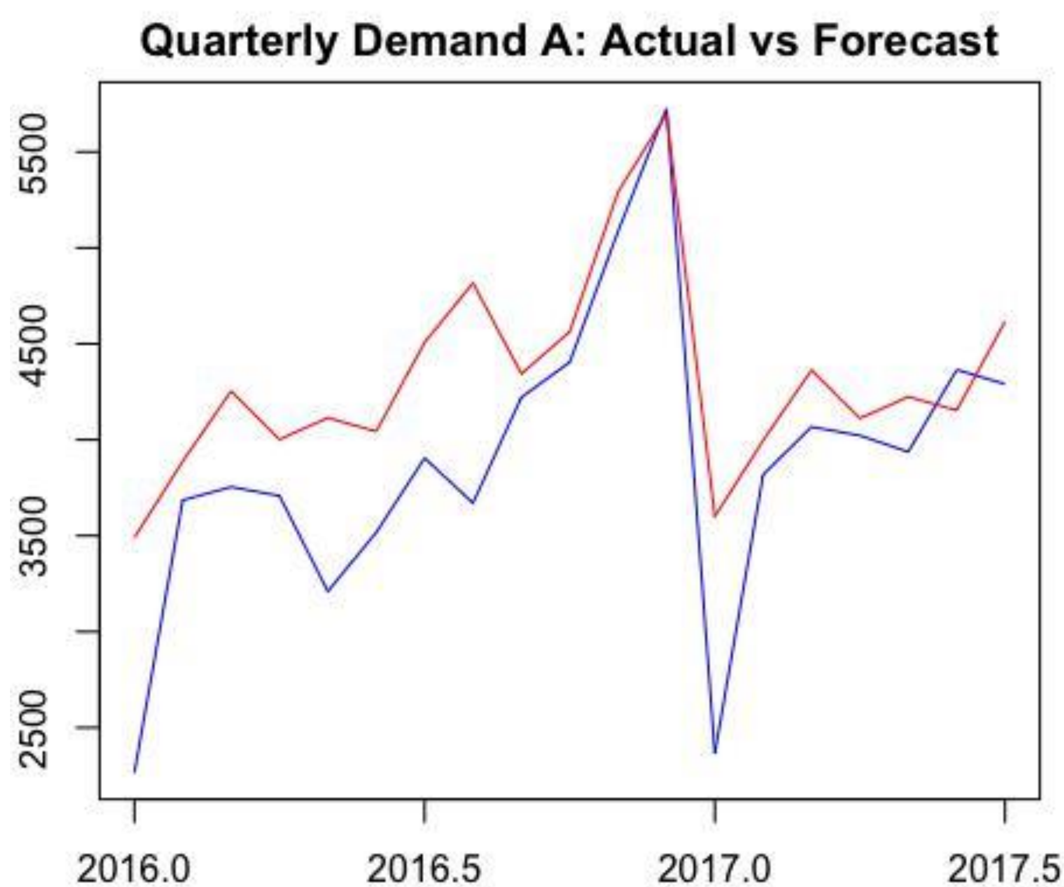
Random Walk with drift model - Forecasting on train data

It assumes that, at each point in time, the series merely takes a random step away from its last recorded position, with steps whose mean value is zero.

```
library(forecast)

## Warning: package 'forecast' was built under R version 3.4.2

fcst.ItA.stl <- forecast(ItmATrn, method="rwdrift", h=19)
fcst.ItB.stl <- forecast(ItmBTrn, method="rwdrift", h=19)

VecA<- cbind(DataATest,fcst.ItA.stl$mean)
VecB<- cbind(DataBTest,fcst.ItB.stl$mean)
```

Item A

```
par(mfrow=c(1,1), mar=c(2, 2, 2, 2), mgp=c(3, 1, 0), las=0)
ts.plot(VecA, col=c("blue", "red"),xlab="year", ylab="demand", main="Quarterl
y Demand A: Actual vs Forecast")
```



Mean absolute percentage error (MAPE)

Calculates the mean absolute percentage error (Deviation) function for the forecast and the eventual outcomes.

```
MAPEA <- mean(abs(VecA[,1]-VecA[,2])/VecA[,1])
MAPEA

## [1] 0.1408798
```

Box-Ljung Test

To check is resuidual are independent

H0: Residuals are independent

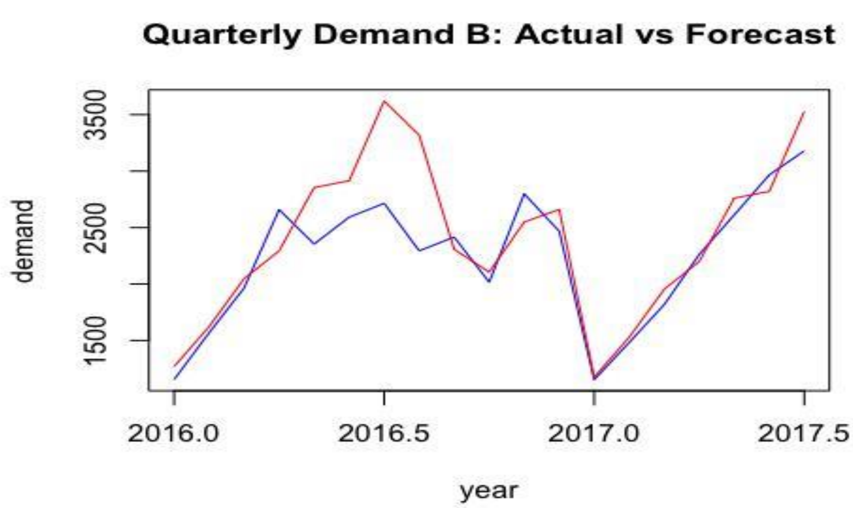
Ha: Residuals are not independent

```
Box.test(fcst.ItA.stl$residuals, lag=30, type="Ljung-Box")

##
##  Box-Ljung test
##
## data:  fcst.ItA.stl$residuals
## X-squared = 167.07, df = 30, p-value < 2.2e-16
```

Conclusion: Reject H0: Residuals are not independent

Item B

```
ts.plot(VecB, col=c("blue", "red"),xlab="year", ylab="demand", main="Quarterl
y Demand B: Actual vs Forecast")
```



```r
MAPEB <- mean(abs(VecB[,1]-VecB[,2])/VecB[,1])
MAPEB

## [1] 0.1082608

Box.test(fcst.ItB.stl$residuals, lag=30, type="Ljung-Box")

##
##  Box-Ljung test
##
## data:  fcst.ItB.stl$residuals
## X-squared = 123.22, df = 30, p-value = 2.931e-13
```

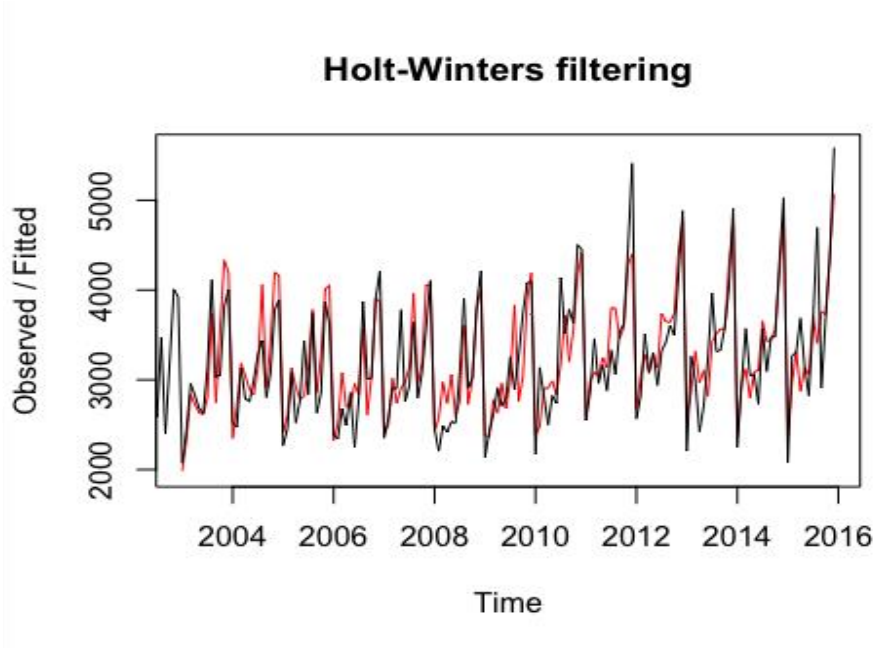
Conclusion: Reject H0: Residuals are not independent

From the above MAPE results we can see the 14 % and 10.8% less accuracy in model.

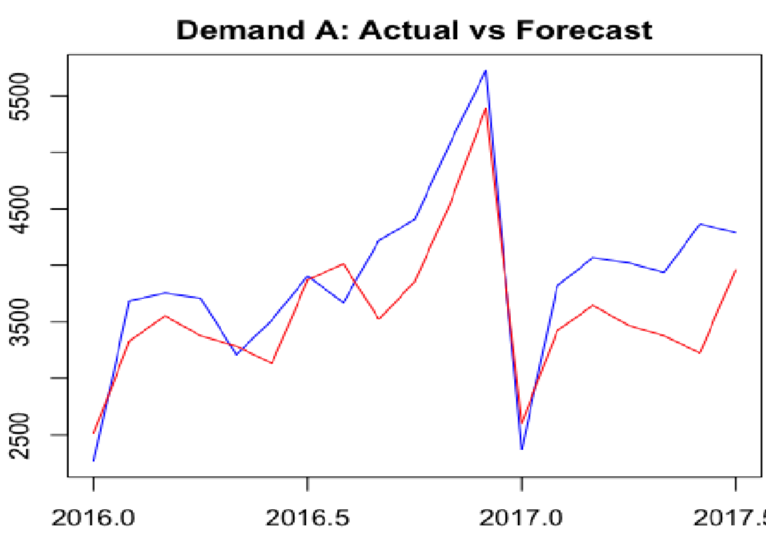
Holt Winter

The Holt-Winters seasonal method comprises the forecast equation and three smoothing equations — one for the level, one for trend, and one for the seasonal component, with smoothing parameters α, β and γ.

```
hwA <- HoltWinters(as.ts(DataATrain),seasonal="additive")
hwA

## Holt-Winters exponential smoothing with trend and additive seasonal
compon ent.
##
## Call:
## HoltWinters(x = as.ts(DataATrain), seasonal = "additive")
##
## Smoothing parameters:
##  alpha: 0.1241357
##  beta : 0.03174654
##  gamma: 0.3636975
##
## Coefficients:
##             [,1]
## a    3753.348040
## b       7.663395
## s1  -1250.098605
## s2   -438.592232
## s3   -224.017731
## s4   -407.395313
## s5   -507.668223
## s6   -667.267246
## s763.659702
## s8197.909330
## s9   -301.525945
## s1025.272325
## s11   712.529546
## s12  1545.291998

hwA$SSE

## [1] 18898609

plot(hwA)
```




```
hwAForecast <- forecast(hwA, h=19)
VecA1 <- cbind(DataATest,hwAForecast)
par(mfrow=c(1,1), mar=c(2, 2, 2, 2), mgp=c(3, 1, 0), las=0)
ts.plot(VecA1[,1],VecA1[,2], col=c("blue","red"),xlab="year", ylab="demand",
main="Demand A: Actual vs Forecast")
```




```
Box.test(hwAForecast$residuals, lag=20, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  hwAForecast$residuals
## X-squared = 14.227, df = 20, p-value = 0.8188
```

Conclusion: Do not reject H0: Residuals are independent

```r
library(MLmetrics)
```

```
##
## Attaching package: 'MLmetrics'

## The following object is masked from 'package:base':
##
##     Recall
```

```r
MAPE(VecA1[,1],VecA1[,2])
```

```
## [1] 0.1160528
```

Item B

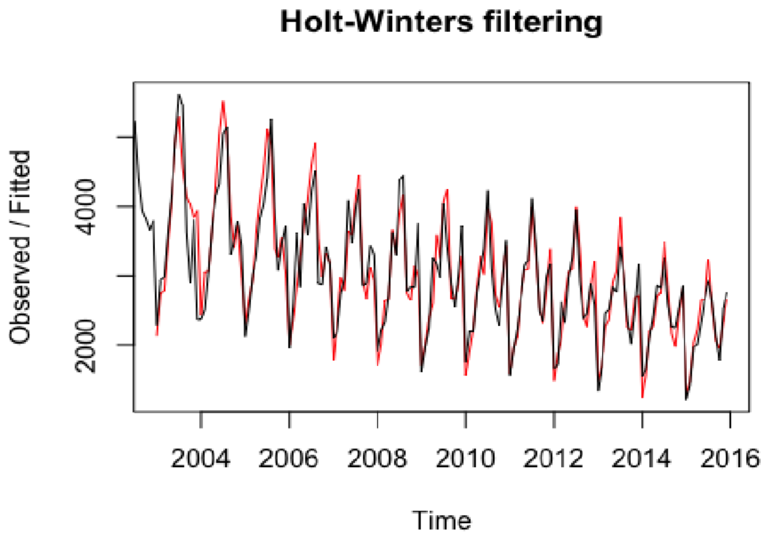
```r
hwB <- HoltWinters(as.ts(DataBTrain),seasonal="additive")
hwB
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal
compon ent.
##
## Call:
## HoltWinters(x = as.ts(DataBTrain), seasonal = "additive")
##
## Smoothing parameters:
##  alpha: 0.0166627
##  beta : 0.4878834
##  gamma: 0.5000132
##
## Coefficients:
##             [,1]
## a    2297.12724
## b     -15.29024
## s1  -1222.01821
## s2  -1012.34884
## s3   -442.56913
## s4   -307.95973
## s579.56065
## s6258.33260
## s7697.64492
## s8241.68337
## s9   -246.12729
## s10  -465.09216
## s11   120.77708
## s12   412.50043
```

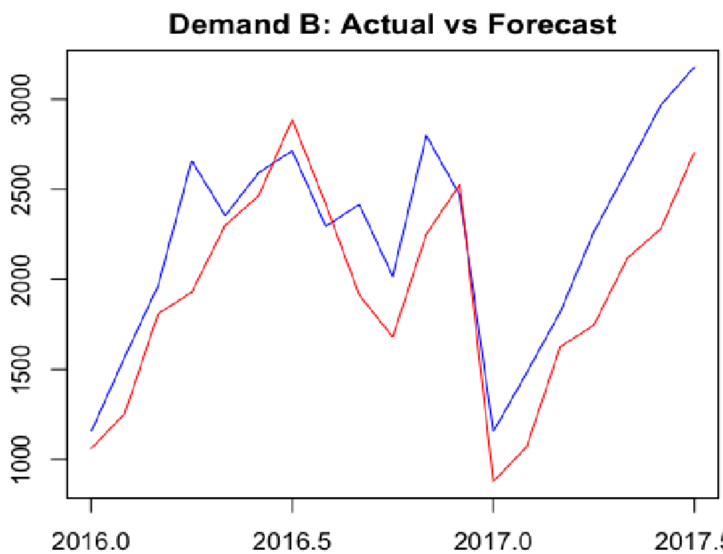
```r
hwB$SSE
```

```
## [1] 17862785
```

```r
plot(hwB)
```




```
hwBForecast <- forecast(hwB, h=19)
VecB1 <- cbind(DataBTest,hwBForecast)
par(mfrow=c(1,1), mar=c(2, 2, 2, 2), mgp=c(3, 1, 0), las=0)
ts.plot(VecB1[,1],VecB1[,2], col=c("blue","red"),xlab="year", ylab="demand",
main="Demand B: Actual vs Forecast")
```




```
Box.test(hwBForecast$residuals, lag=20, type="Ljung-Box")

##
##  Box-Ljung test
##
## data:  hwBForecast$residuals
## X-squared = 13.101, df = 20, p-value = 0.873
```

Conclusion: Do not reject H0: Residuals are independent

```r
MAPE(VecB1[,1],VecB1[,2])

## [1] 0.1867152
```

MAPE is 11.6 % and 18.6 % for item A and Item B resp.

Check for stationary time series

Dickey–Fuller test

Statistical tests make strong assumptions about your data. They can only be used to inform the degree to which a null hypothesis can be accepted or rejected. The result must be interpreted for a given problem to be meaningful. Nevertheless, they can provide a quick check and confirmatory evidence that your time series is stationary or non-stationary.

Null Hypothesis (H0): If accepted, it suggests the time series has a unit root, meaning it is non-stationary. It has some time dependent structure.

Alternate Hypothesis (H1): The null hypothesis is rejected; it suggests the time series does not have a unit root, meaning it is stationary. It does not have time-dependent structure.

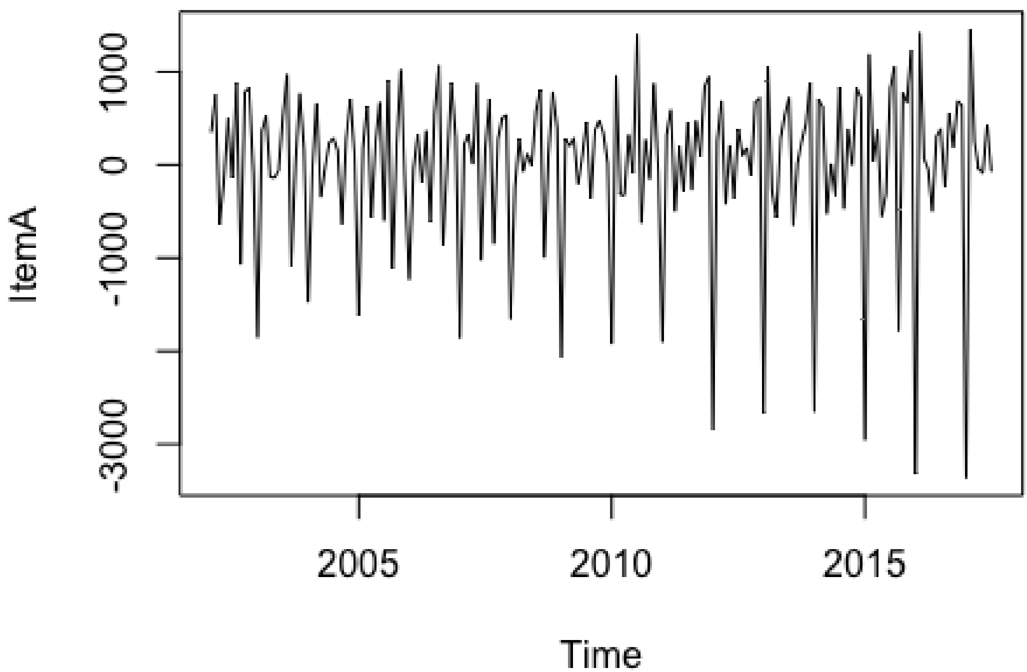
p-value > 0.05: Accept the null hypothesis (H0), the data has a unit root and is non-stationary. p-value <= 0.05: Reject the null hypothesis (H0), the data does not have a unit root and is stationary.

Item A

```
library(tseries)
adf.test(dem_ItA)

## Warning in adf.test(dem_ItA): p-value smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  dem_ItA
## Dickey-Fuller = -7.8632, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

Returns suitably lagged and iterated differences.

```
diff_dem_ItA <- diff(dem_ItA)
plot(diff_dem_ItA)
```

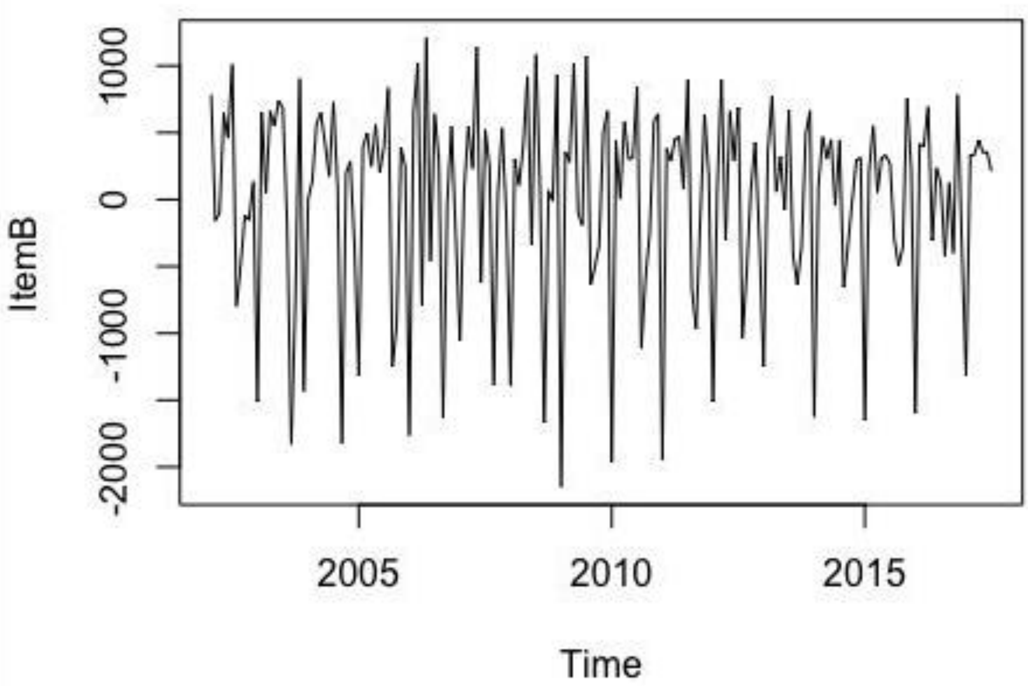



```
adf.test(diff(dem_ItA))

## Warning in adf.test(diff(dem_ItA)): p-value smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  diff(dem_ItA)
## Dickey-Fuller = -8.0907, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

Item B

```
adf.test(dem_ItB)

## Warning in adf.test(dem_ItB): p-value smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  dem_ItB
## Dickey-Fuller = -12.967, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

```r
diff_dem_ItB <- diff(dem_ItB)
plot(diff_dem_ItB)
```




```r
adf.test(diff(dem_ItB))
```

```
## Warning in adf.test(diff(dem_ItB)): p-value smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  diff(dem_ItB)
## Dickey-Fuller = -9.8701, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

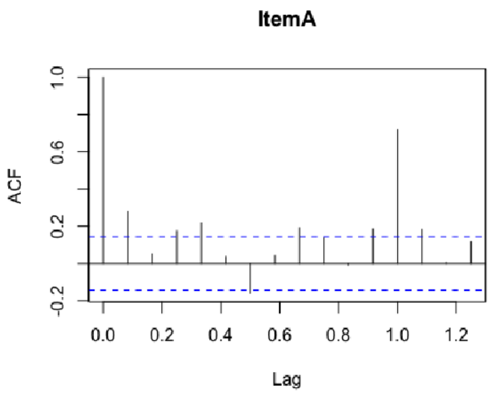
From the above ADF test the Null Hypothesis is rejected. The time series of differences (above) does appear to be stationary in mean and variance, as the level of the series stays roughly constant over time, and the variance of the series appears roughly constant over time.

ACF and PACF (performing to check the stationary data and autocorrelation)

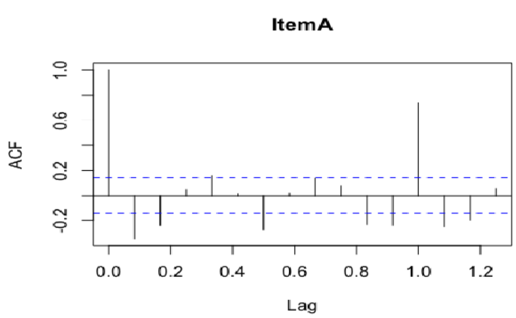
The function Acf computes an estimate of the autocorrelation function of a (possibly multivariate) time series. Function Pacf computes an estimate of the partial autocorrelation function of a (possibly multivariate) time series.

You can difference a time series using the “diff()” function in R Checking with Lag 15

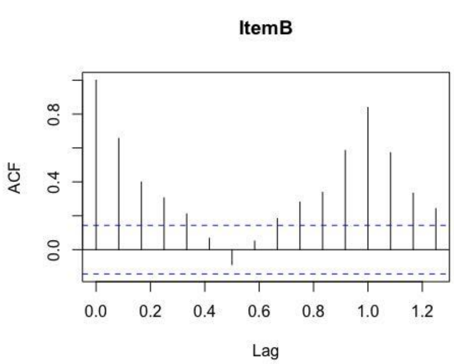
```
acf(dem_ItA,lag=15)
```



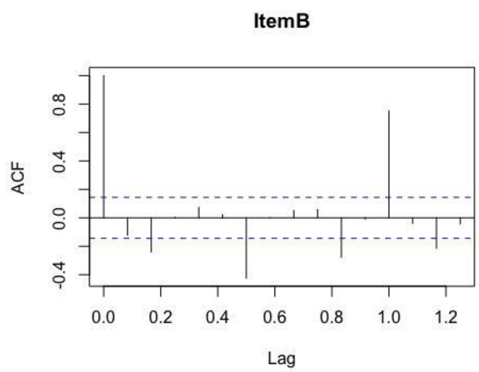

```
acf(diff_dem_ItA, lag=15)
```




```
acf(dem_ItB,lag=15)
```

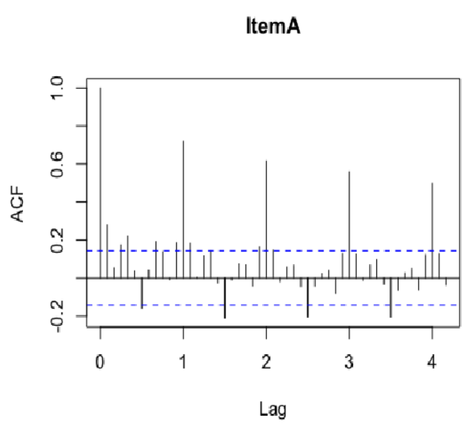


```
acf(diff_dem_ItB, lag=15)
```

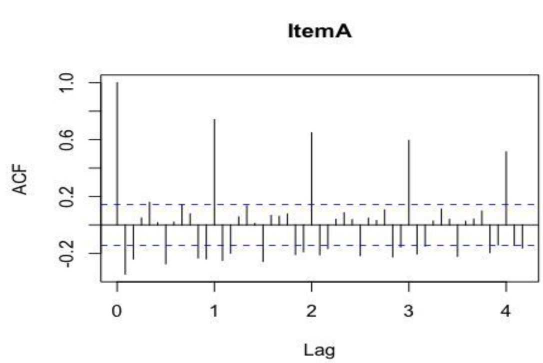


Checking with Lag 50

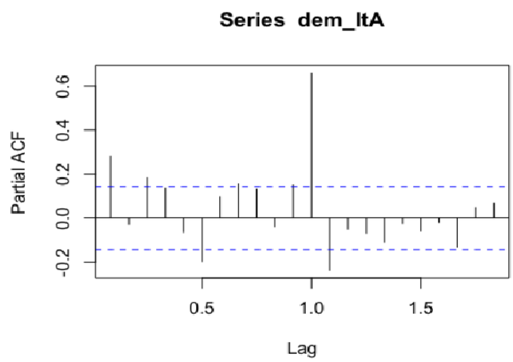
```
acf(dem_ItA,lag=50)
```



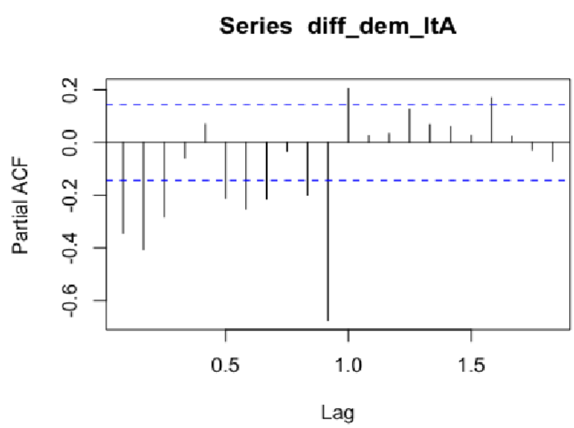

```
acf(diff_dem_ItA, lag=50)
```



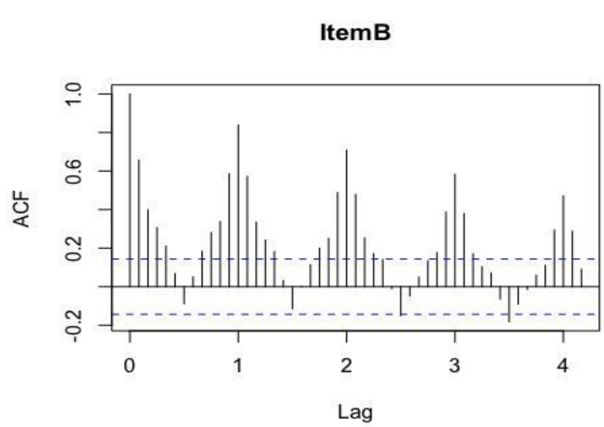
```
pacf(dem_ItA)
```



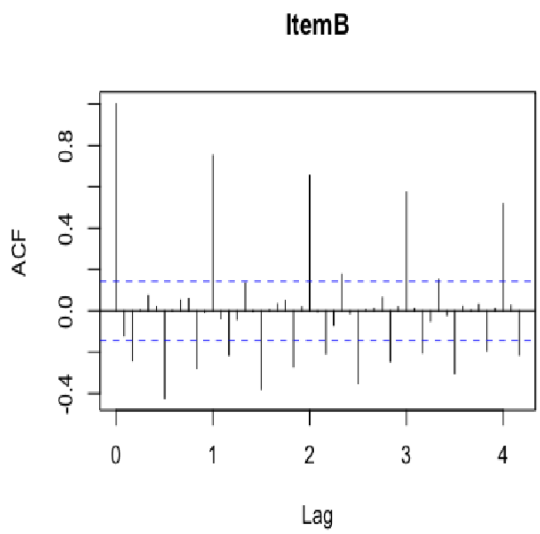

```
pacf(diff_dem_ItA)
```



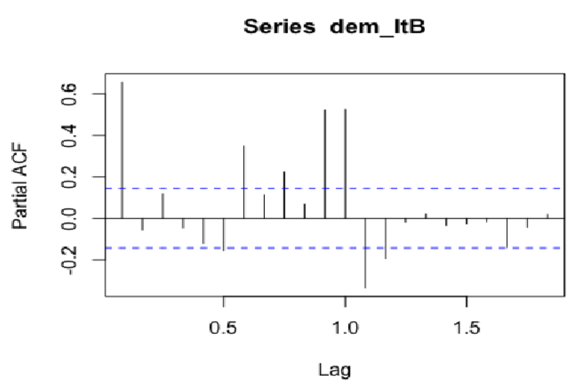

```
acf(dem_ItB,lag=50)
```



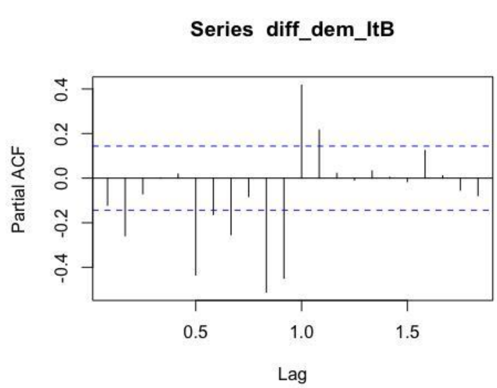

```
acf(diff_dem_ItB, lag=50)
```




```
pacf(dem_ItB)
```




```
pacf(diff_dem_ItB)
```



ARMA model

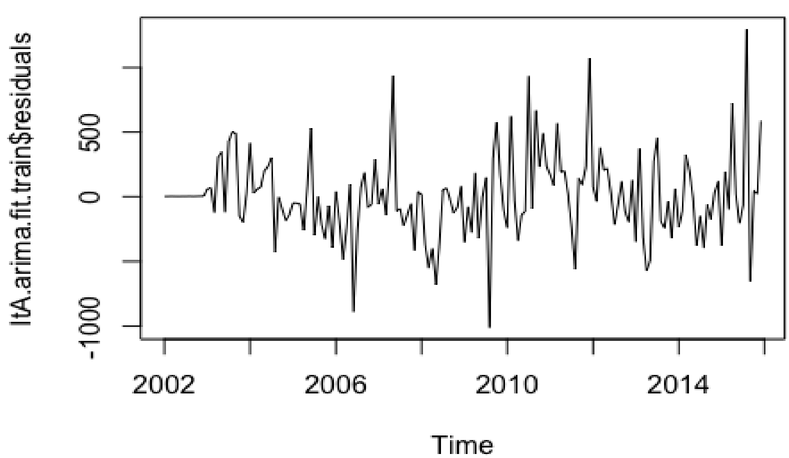
ARMA models are commonly used in time series modeling. In ARMA model, AR stands for auto-regression and MA stands for moving average. ARMA model is performed on non-stationary data. In this case value are stationary we cannot perform ARMA model. Also, from above ACF and PACF we have found out that the positive and negative values mean (that is because of data is stationary; there are not cuts for AR(2) series and no gradually decrease in the value of PACF, no significance of MA(2).

ARIMA Model

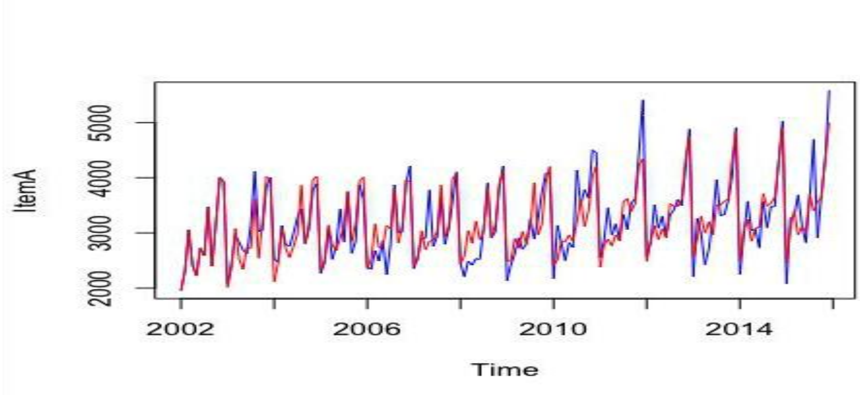
Exponential smoothing methods are useful for making forecasts, and make no assumptions about the correlations between successive values of the time series. While exponential smoothing methods do not make any assumptions about correlations between successive values of the time series, in some cases you can make a better predictive model by taking correlations in the data into account. Autoregressive Integrated Moving Average (ARIMA) models include an explicit statistical model for the irregular component of a time series, that allows for non-zero autocorrelations in the irregular component. ARIMA models are defined for stationary time series.

Item A

```
ItA.arima.fit.train <- auto.arima(DataATrain, seasonal=TRUE)
ItA.arima.fit.train

## Series: DataATrain
## ARIMA(0,0,0)(0,1,1)[12] with drift
##
## Coefficients:
##          sma1   drift
##       -0.6581  3.9132
## s.e.   0.0798  0.9188
##
## sigma^2 estimated as 116022:  log likelihood=-1133.35
## AIC=2272.71   AICc=2272.86   BIC=2281.86

plot(ItA.arima.fit.train$residuals)
```



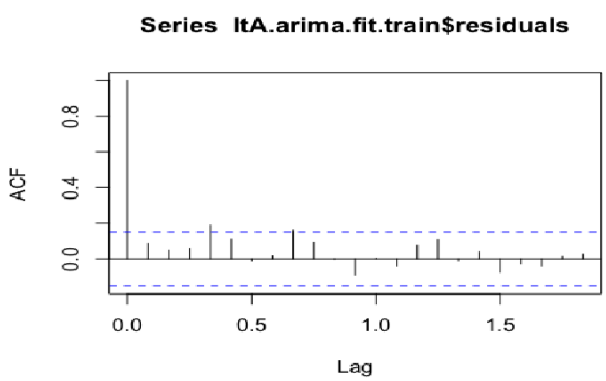

```
plot(ItA.arima.fit.train$x,col="blue")
lines(ItA.arima.fit.train$fitted,col="red",main="Demand A: Actual vs Forecast
")
```



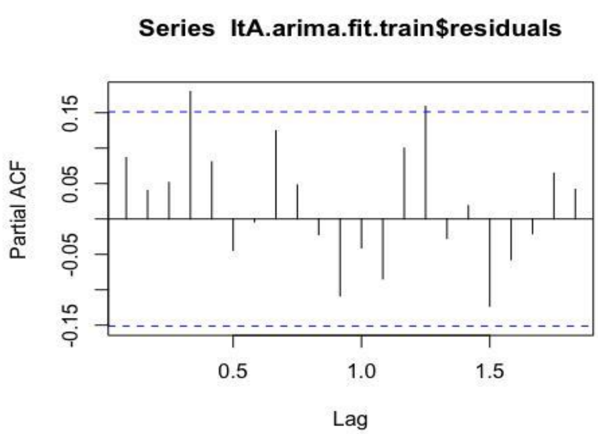

```
MAPE(ItA.arima.fit.train$fitted,ItA.arima.fit.train$x)
## [1] 0.0733376
```

We can meas percentage error is now reduced to 7.3% for ARIMA.

```
acf(ItA.arima.fit.train$residuals)
```




```
pacf(ItA.arima.fit.train$residuals)
```



Box-Ljung Test

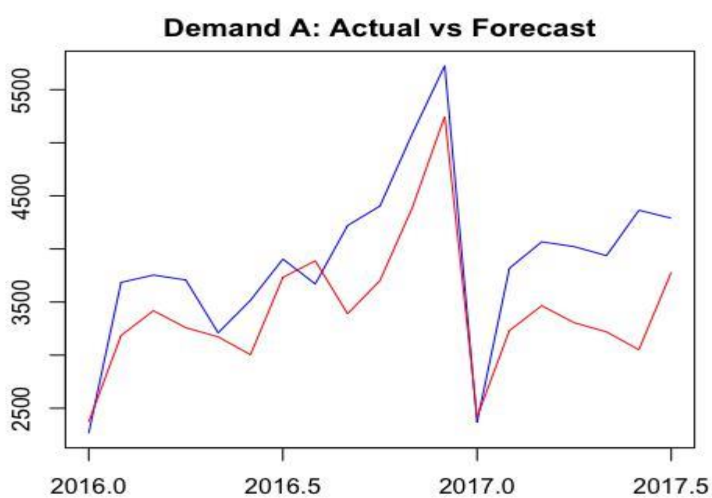
To check is resuidual are independent H0: Residuals are independent Ha: Residuals are not independent

```
Box.test(ItA.arima.fit.train$residuals, lag = 30, type = c("Ljung-Box"), fitd
f = 0)

##
##  Box-Ljung test
##
## data:  ItA.arima.fit.train$residuals
## X-squared = 33.158, df = 30, p-value = 0.3157
```

Conclusion: Do not reject H0: Residuals are independent

Forecasting on hold dataset

```
ArimafcastA <- forecast(ItA.arima.fit.train, h=19)
VecA2 <- cbind(DataATest,ArimafcastA)
par(mfrow=c(1,1), mar=c(2, 2, 2, 2), mgp=c(3, 1, 0), las=0)
ts.plot(VecA2[,1],VecA2[,2], col=c("blue","red"),xlab="year", ylab="demand",
main="Demand A: Actual vs Forecast")
```



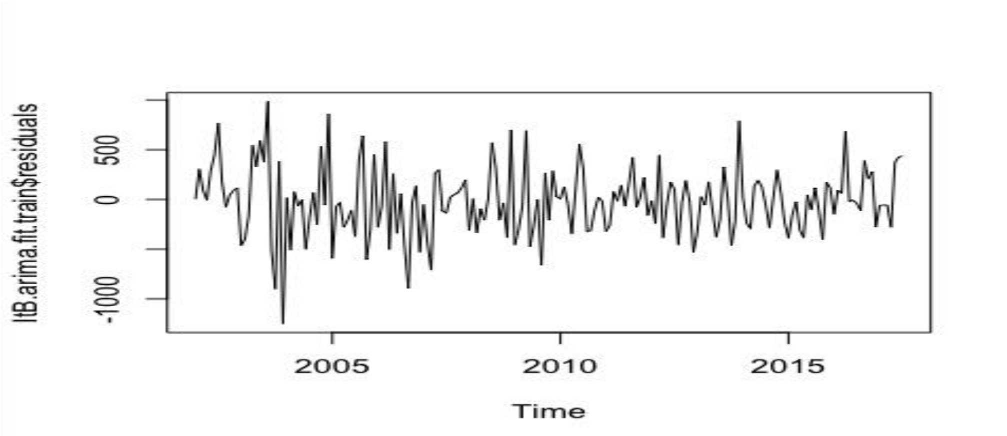
From the plot and data, we can see the forecasted value for follows almost the same as actual value, there are point of interaction at Jan 2016, May 2016, Dec 2016, Jan 2017.

Item B

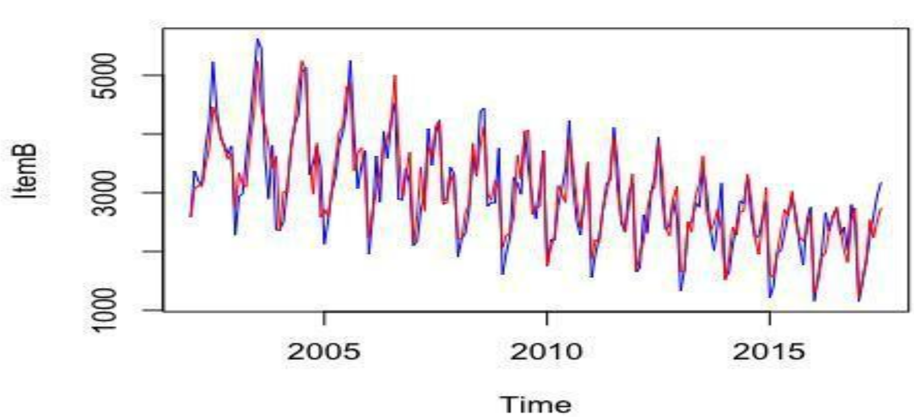
```
ItB.arima.fit.train <- auto.arima(dem_ItB, seasonal=TRUE)
ItB.arima.fit.train
```

```
## Series: dem_ItB
## ARIMA(4,1,1)(1,0,0)[12]
##
## Coefficients:
##           ar1     ar2      ar3      ar4      ma1   sar1
##        0.1516  0.0892  -0.0332  -0.1433  -0.9652 0.8600
## s.e.   0.0760  0.0746   0.0746   0.0753   0.0161 0.0339
##
## sigma^2 estimated as 121950:  log likelihood=-1358.6
## AIC=2731.2   AICc=2731.83   BIC=2753.78
```

```
plot(ItB.arima.fit.train$residuals)
```



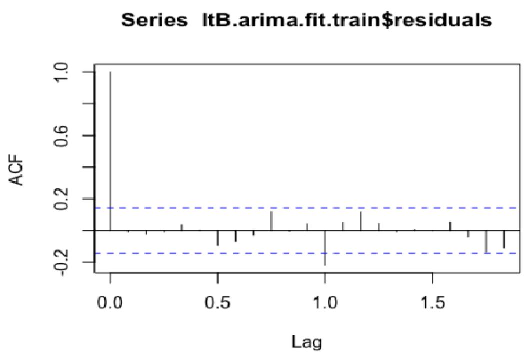

```
plot(ItB.arima.fit.train$x,col="blue")
lines(ItB.arima.fit.train$fitted,col="red", main="Demand B: Actual vs Forecas
t")
```



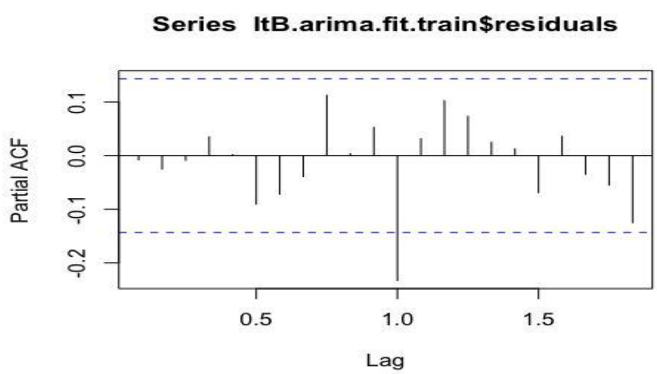
```r
MAPE(ItB.arima.fit.train$fitted,ItB.arima.fit.train$x)
## [1] 0.09087366
```

We can meas percentage error is now reduced to 9% for ARIMA.

```r
acf(ItB.arima.fit.train$residuals)
```




```r
pacf(ItB.arima.fit.train$residuals)
```



Box-Ljung Test

To check is residual are independent

H0: Residuals are independent

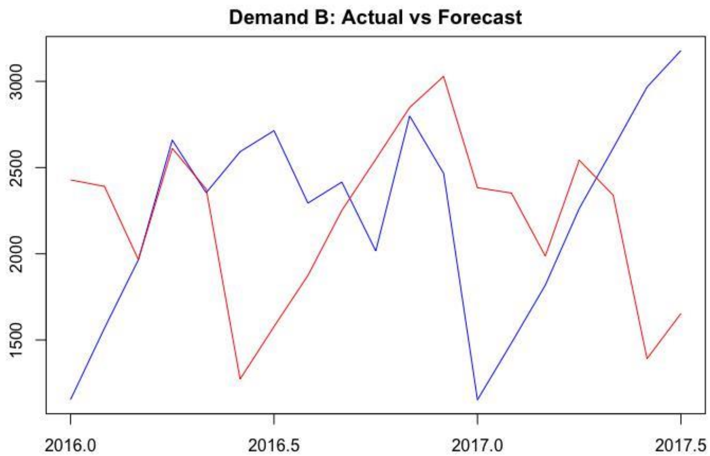
Ha: Residuals are not independent

```r
Box.test(ItB.arima.fit.train$residuals, lag = 30, type = c("Ljung-Box"), fitd
f = 0)
##
## 	Box-Ljung test
##
## data:  ItB.arima.fit.train$residuals
## X-squared = 37.735, df = 30, p-value = 0.1567
```

Conclusion: Do not reject H0: Residuals are independent

Forecasting on hold dataset

```
ArimafcastB <- forecast(ItB.arima.fit.train, h=19)
VecB2 <- cbind(DataBTest,ArimafcastB)
par(mfrow=c(1,1), mar=c(2, 2, 2, 2), mgp=c(3, 1, 0), las=0)
ts.plot(VecB2[,1],VecB2[,2], col=c("blue","red"),xlab="year", ylab="demand",
main="Demand B: Actual vs Forecast")
```



From the plot and data, we can see the forecasted value doesn't exactly follows the actual value, but there are point of interaction at Mar 2016, Apr 2016, May 2016 Nov 2016, Mar 2017.

Conclusion

For Time Series Forecasting problem, we had observed the, trend and seasonality in the data. We have observed the Item A has increasing trend, but for Item B the trend is declining. Also, we observed for both item there are few months with high variation in seasonality; and for Item A there are few outliers.

As the seasonality was not following the trend pattern we have used the “Additive” seasonality. We have performed the three models Random Walk with Drift, Holt Winters and ARIMA model. As the data was stationary we haven’t used the ARMA model. Below are MAPE and Box-Ljung test observations for Models.

Random Walk with Drift

Item A# 0.1408798 (14%), p-value < 2.2e-16

Item B# 0.1082608 (10.8%), p-value = 2.931e-13

Holt Winters

Item A# 0.1160528 (11.6%), p-value = 0.8188

Item B# 0.1867152 (18.6%), p-value = 0.873

ARIMA

Item A# 0.0733376 (7%), p-value = 0.3157

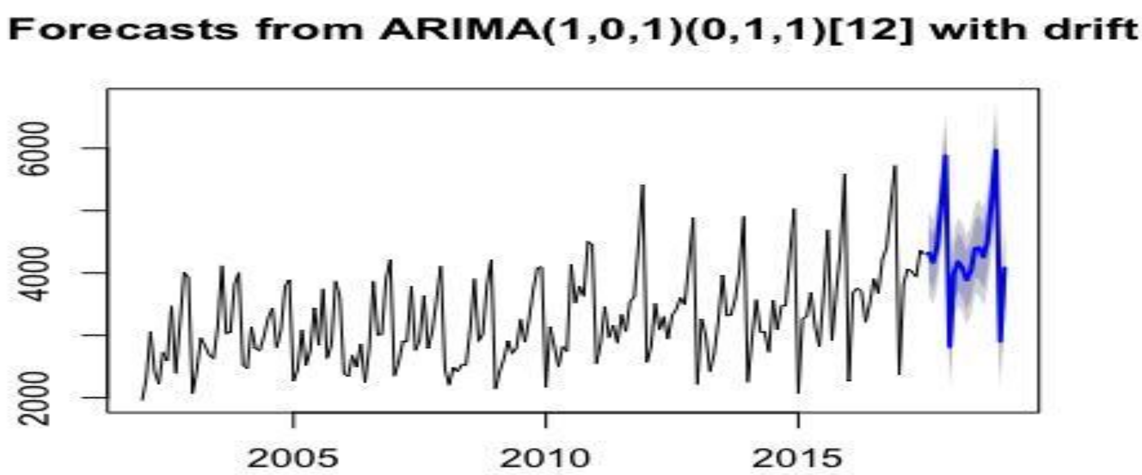
Item B# 0.09087366 (9%), p-value = 0.1567

From the MAPE values observed the ARIMA model provided the lowest values and we selected the model for the Forecasting.

Forecasting using ARIMA Model

Item A

```
ItA.arima.fit <- auto.arima(dem_ItA, seasonal=TRUE)
fcastA <- forecast(ItA.arima.fit, h=19)
plot(fcastA)
```



Item B

```
ItB.arima.fit <- auto.arima(dem_ItB, seasonal=TRUE)
fcastB <- forecast(ItB.arima.fit, h=19)
plot(fcastB)
```