Chulwalhar Export Forecast - Unit10CaseStudy

Paola Leon, Eyal Greenberg and Kyle Killion

July 11, 2016

Introduction

The prime minister of Chulwalhar has asked us to help him in forecasting exports from his country. In order to do this we have been given as-is data, which is the original or observed data, and planned data, which is what Chulwalhar would like to export. We also have a list of indicators that may affect exports. Our job is to find out the best way to forecast Chulwalhar's exports in 2014 based on data collected before this year. In other words, we want to create a credible statistical model.

The export data for Chulwalhar are in two CSV files. One contains the as-is data and the other one contains the planned data. These data sets are also composed of other data sets: monthly and yearly for both groups. Your task is to take all of these data sets, import them into R, and develop a model to forecast the exports of these particular products for the prime minister of Chulwalhar.

Original data sources can be found in the link below:

https://s3-us-west-2.amazonaws.com/smu-mds/prod/MSDS+6306+Doing+Data+Science/Week+10/ChulwalarCase.zip (https://s3-us-west-2.amazonaws.com/smu-mds/prod/MSDS+6306+Doing+Data+Science/Week+10/ChulwalarCase.zip)

Instructions

1. Submit an R markdown document with the code necessary to download, clean and analyze the data.
2. Interpretations of the code and analysis should be provided.
3. There should also be at least one graphic that explains an important feature of the data.
4. This graphic should be interpreted in the text of the document.

Recording the session info

```
sessionInfo()
```

```
## R version 3.2.4 (2016-03-10)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: OS X 10.11.4 (El Capitan)
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] magrittr_1.5    formatR_1.3     tools_3.2.4     htmltools_0.3.5
##  [5] yaml_2.1.13     Rcpp_0.12.4     stringi_1.0-1   rmarkdown_0.9.6
##  [9] knitr_1.12.3    stringr_1.0.0   digest_0.6.9    evaluate_0.9
```

Set the working directory

```
setwd("/Users/paolaleon/Desktop/MSDS 6303 Doing Data Science/Unit10CaseStudy")
getwd()
```

```
## [1] "/Users/paolaleon/Desktop/MSDS 6303 Doing Data Science/Unit10CaseStudy"
```

Download datasources.

1. Manually clicked in the link above and downloaded a .zip file name **ChulwalarCase.zip**
2. Manually unzipped the file and placed its contents under the working directory (in our local machine).
3. Using R, accessed the .csv files as follows:

```
list.files()
```

```
## [1] "ChulwalarCase.zip"
## [2] "Forecasting Exports Chulwalar_0.8a.R"
## [3] "ImportedAsIsDataChulwalar.csv"
## [4] "ImportedIndicatorsChulwalar.csv"
## [5] "ImportedPlanDataChulwalar.csv"
## [6] "Instructions.txt"
## [7] "PLeon_EGreenberg_KKillion_Unit10CaseStudy.html"
## [8] "PLeon_EGreenberg_KKillion_Unit10CaseStudy.Rmd"
```

```
rawImportedAsIsDataChulwalar <- read.csv("ImportedAsIsDataChulwalar.csv", sep = ";",
stringsAsFactors=FALSE, header=TRUE)
rawImportedPlanDataChulwalar <- read.csv("ImportedPlanDataChulwalar.csv", sep = ";",
stringsAsFactors=FALSE, header=TRUE)
rawImportedIndicatorsChulwalar <- read.csv("ImportedIndicatorsChulwalar.csv", sep = "
;", stringsAsFactors=FALSE, header=TRUE)
```

Analysis of raw data

1. General description of the dataframes: number of observations, number of variables, datatypes, and existance of NAs.

```
str(rawImportedAsIsDataChulwalar)
```

```
## 'data.frame':    97 obs. of  8 variables:
##  $ Total.As.Is: chr  "Jan" "Feb" "Mar" "Apr" ...
##  $ X2008      : int  2313221 1950131 2346635 2039787 1756964 1458302 1679637 16396
70 2882886 2959716 ...
##  $ X2009      : int  2610573 2371327 2743786 2125308 1850073 1836222 1797311 18519
68 3271171 2818888 ...
##  $ X2010      : int  2760688 2918333 3227041 1613888 2550157 2317645 1474144 21485
21 3898571 3348953 ...
##  $ X2011      : int  3112861 2926663 3294784 2577079 2774068 2378227 2222900 29917
87 4151531 3318684 ...
##  $ X2012      : int  3093088 3679308 3433364 2714899 3011767 2726028 2483834 30556
55 4200796 4228724 ...
##  $ X2013      : int  4119526 3535744 3560974 3760065 2959933 2787898 2828744 30841
13 5107775 4562144 ...
##  $ X2014      : int  4308161 4155378 3924332 3659121 3898758 3313891 3595106 35024
26 5619059 5274287 ...
```

```
str(rawImportedPlanDataChulwalar)
```

```
## 'data.frame':    96 obs. of  8 variables:
##  $ Total.Plan: chr  "Jan" "Feb" "Mar" "Apr" ...
##  $ X2008     : int  2243103 2162705 2720911 2011182 1877757 1819924 1682196 189317
1 3325711 2662148 ...
##  $ X2009     : int  2547980 2247049 2731156 2020158 2098038 1927995 1783692 190770
5 3124040 3102251 ...
##  $ X2010     : int  2965885 2751170 2906493 2383358 2246893 1992851 2023434 224499
7 3257717 3536338 ...
##  $ X2011     : int  3113110 2883766 2957893 2601648 2370949 2339881 2105328 234162
3 4086297 3640827 ...
##  $ X2012     : int  3895396 3588151 3787240 3036434 2907891 2707822 2619486 378455
7 4987460 4367319 ...
##  $ X2013     : int  3580325 3863212 3606083 3213575 3139128 2998610 2785453 308365
4 5143757 4149334 ...
##  $ X2014     : int  4474000 4185565 4278119 3985542 3605973 3515173 3269444 365611
2 5637391 5157781 ...
```

```
str(rawImportedIndicatorsChulwalar)
```

```
## 'data.frame':    194 obs. of  8 variables:
##  $ Change.in.export.prices: chr  "Jan" "Feb" "Mar" "Apr" ...
##  $ X2008                  : num  97.4 97.8 98.3 98.1 98.7 98.9 99.5 99.2 99.1 98.9
...
##  $ X2009                  : num  98.3 98.9 98.7 98.8 98.7 99 99 99.2 98.9 98.9 ...
##  $ X2010                  : num  99 99.4 99.9 100 99.9 ...
##  $ X2011                  : num  101 101 102 102 102 ...
##  $ X2012                  : num  103 104 104 104 104 ...
##  $ X2013                  : num  104 105 106 105 106 ...
##  $ X2014                  : num  NA NA NA NA NA NA NA NA NA NA ...
```

2. A snapshot of the dataframes: top six rows

```
head(rawImportedAsIsDataChulwalar)
```

```
##   Total.As.Is   X2008   X2009   X2010   X2011   X2012   X2013   X2014
## 1         Jan 2313221 2610573 2760688 3112861 3093088 4119526 4308161
## 2         Feb 1950131 2371327 2918333 2926663 3679308 3535744 4155378
## 3         Mar 2346635 2743786 3227041 3294784 3433364 3560974 3924332
## 4         Apr 2039787 2125308 1613888 2577079 2714899 3760065 3659121
## 5         May 1756964 1850073 2550157 2774068 3011767 2959933 3898758
## 6         Jun 1458302 1836222 2317645 2378227 2726028 2787898 3313891
```

```
head(rawImportedPlanDataChulwalar)
```

```
##   Total.Plan   X2008   X2009   X2010   X2011   X2012   X2013   X2014
## 1        Jan 2243103 2547980 2965885 3113110 3895396 3580325 4474000
## 2        Feb 2162705 2247049 2751170 2883766 3588151 3863212 4185565
## 3        Mar 2720911 2731156 2906493 2957893 3787240 3606083 4278119
## 4        Apr 2011182 2020158 2383358 2601648 3036434 3213575 3985542
## 5        May 1877757 2098038 2246893 2370949 2907891 3139128 3605973
## 6        Jun 1819924 1927995 1992851 2339881 2707822 2998610 3515173
```

```
head(rawImportedIndicatorsChulwalar)
```

```
##   Change.in.export.prices X2008 X2009 X2010 X2011 X2012 X2013 X2014
## 1                     Jan  97.4  98.3  99.0 100.7 102.8 104.5    NA
## 2                     Feb  97.8  98.9  99.4 101.3 103.5 105.1    NA
## 3                     Mar  98.3  98.7  99.9 101.9 104.1 105.6    NA
## 4                     Apr  98.1  98.8 100.0 101.9 103.9 105.1    NA
## 5                     Mai  98.7  98.7  99.9 101.9 103.9 105.5    NA
## 6                     Jun  98.9  99.0  99.9 102.0 103.7 105.6    NA
```

3. A snapshot of the dataframes: last six rows

```
tail(rawImportedAsIsDataChulwalar)
```

```
##    Total.As.Is    X2008    X2009    X2010    X2011    X2012    X2013 X2014
## 92         Aug 26280011 29609916 32726772 37215503 40629676 45408410    NA
## 93         Sep 26280011 29609916 32726772 37215503 40629676 45408410    NA
## 94         Oct 26280011 29609916 32726772 37215503 40629676 45408410    NA
## 95         Nov 26280011 29609916 32726772 37215503 40629676 45408410    NA
## 96         Dez 26280011 29609916 32726772 37215503 40629676 45408410    NA
## 97                   NA       NA       NA       NA       NA       NA    NA
```

```
tail(rawImportedPlanDataChulwalar)
```

```
##    Total.Plan    X2008    X2009    X2010    X2011    X2012    X2013 X2014
## 91        Jul 27883407 29387100 32780247 35224132 43947063 44152007    NA
## 92        Aug 27883407 29387100 32780247 35224132 43947063 44152007    NA
## 93        Sep 27883407 29387100 32780247 35224132 43947063 44152007    NA
## 94        Oct 27883407 29387100 32780247 35224132 43947063 44152007    NA
## 95        Nov 27883407 29387100 32780247 35224132 43947063 44152007    NA
## 96        Dec 27883407 29387100 32780247 35224132 43947063 44152007    NA
```

```
tail(rawImportedIndicatorsChulwalar)
```

```
##     Change.in.export.prices X2008 X2009 X2010 X2011 X2012 X2013 X2014
## 189                     Jul     0     0     0     0     0     0     0
## 190                     Aug     0     0     0     0     0     0     0
## 191                     Sep     1     1     1     1     1     1     1
## 192                     Oct     0     0     0     0     0     0     0
## 193                     Nov     1     1     1     1     1     1     1
## 194                     Dec     1     1     1     1     1     1     1
```

4. Identify the column headers:

```
names(rawImportedAsIsDataChulwalar)
```

```
## [1] "Total.As.Is" "X2008"       "X2009"       "X2010"       "X2011"
## [6] "X2012"       "X2013"       "X2014"
```

```
names(rawImportedPlanDataChulwalar)
```

```
## [1] "Total.Plan" "X2008"      "X2009"      "X2010"      "X2011"
## [6] "X2012"      "X2013"      "X2014"
```

```
names(rawImportedIndicatorsChulwalar)
```

```
## [1] "Change.in.export.prices" "X2008"
## [3] "X2009"                   "X2010"
## [5] "X2011"                   "X2012"
## [7] "X2013"                   "X2014"
```

5. Identify the dimensions of the dataframes (number of observations and variables):

```
dim(rawImportedAsIsDataChulwalar)
```

```
## [1] 97  8
```

```
dim(rawImportedPlanDataChulwalar)
```

```
## [1] 96  8
```

```
dim(rawImportedIndicatorsChulwalar)
```

```
## [1] 194   8
```

6. Calculate dataframe summaries for both continuos(numeric) and categorical fields:

```
summary(rawImportedAsIsDataChulwalar)
```

```
##  Total.As.Is            X2008              X2009
##  Length:97          Min.   :    2008   Min.   :    2009
##  Class :character   1st Qu.:  385866   1st Qu.:  403007
##  Mode  :character   Median :  627705   Median :  709129
##                     Mean   : 4232857   Mean   : 4766375
##                     3rd Qu.: 1706358   3rd Qu.: 1851494
##                     Max.   :26280011   Max.   :29609916
##                     NA's   :7          NA's   :7
##      X2010              X2011              X2012
##  Min.   :    2010   Min.   :    2011   Min.   :    2012
##  1st Qu.:  441319   1st Qu.:  573526   1st Qu.:  612274
##  Median :  786690   Median :  942442   Median : 1023916
##  Mean   : 5259622   Mean   : 5950923   Mean   : 6507372
##  3rd Qu.: 2112990   3rd Qu.: 2359488   3rd Qu.: 2657133
##  Max.   :32726772   Max.   :37215503   Max.   :40629676
##  NA's   :7          NA's   :7          NA's   :7
##      X2013              X2014
##  Min.   :    2013   Min.   :3313891
##  1st Qu.:  718698   1st Qu.:3643117
##  Median : 1098900   Median :4039855
##  Mean   : 7276803   Mean   :4229756
##  3rd Qu.: 2927136   3rd Qu.:4709064
##  Max.   :45408410   Max.   :5619059
##  NA's   :7          NA's   :85
```

```
summary(rawImportedPlanDataChulwalar)
```

```
##   Total.Plan            X2008              X2009
##  Length:96          Min.   :    2008   Min.   :    2009
##  Class :character   1st Qu.:  444526   1st Qu.:  400904
##  Mode  :character   Median :  665004   Median :  792752
##                     Mean   : 4496826   Mean   : 4746464
##                     3rd Qu.: 1863299   3rd Qu.: 1922922
##                     Max.   :27883407   Max.   :29387100
##                     NA's   :6          NA's   :6
##      X2010              X2011              X2012
##  Min.   :    2010   Min.   :    2011   Min.   :    2012
##  1st Qu.:  467315   1st Qu.:  556745   1st Qu.:  659396
##  Median :  872645   Median :  833366   Median : 1149886
##  Mean   : 5270048   Mean   : 5661809   Mean   : 7045793
##  3rd Qu.: 2083550   3rd Qu.: 2341188   3rd Qu.: 2794860
##  Max.   :32780247   Max.   :35224132   Max.   :43947063
##  NA's   :6          NA's   :6          NA's   :6
##      X2013              X2014
##  Min.   :    2013   Min.   :3269444
##  1st Qu.:  689432   1st Qu.:3643577
##  Median : 1086328   Median :4231842
##  Mean   : 7075145   Mean   :4318479
##  3rd Qu.: 2977986   3rd Qu.:4816834
##  Max.   :44152007   Max.   :5637391
##  NA's   :6          NA's   :84
```

```
summary(rawImportedIndicatorsChulwalar)
```

```
##  Change.in.export.prices     X2008              X2009
##  Length:194              Min.   :    -25   Min.   :    -32
##  Class :character        1st Qu.:      2   1st Qu.:      1
##  Mode  :character        Median :    100   Median :     99
##                          Mean   : 395192   Mean   : 391910
##                          3rd Qu.:   4988   3rd Qu.:   4338
##                          Max.   :5850000   Max.   :5800000
##                          NA's   :13        NA's   :13
##      X2010              X2011              X2012              X2013
##  Min.   :    -18   Min.   :     -4   Min.   :    -10   Min.   :     -7
##  1st Qu.:      2   1st Qu.:      5   1st Qu.:      2   1st Qu.:      2
##  Median :    100   Median :    102   Median :    105   Median :    106
##  Mean   : 406964   Mean   : 448422   Mean   : 475054   Mean   : 509129
##  3rd Qu.:   5925   3rd Qu.:   5898   3rd Qu.:   6459   3rd Qu.:   7795
##  Max.   :6020000   Max.   :6640000   Max.   :7040000   Max.   :7550000
##  NA's   :13        NA's   :13        NA's   :13        NA's   :13
##      X2014
##  Min.   :      0
##  1st Qu.:      1
##  Median :    219
##  Mean   :1076172
##  3rd Qu.:  61579
##  Max.   :7910000
##  NA's   :105
```

Tidyng Data

1. Create new dataframes for tyding data so that raw data stays untouched.

```
tidyImportedAsIsDataChulwalar <- rawImportedAsIsDataChulwalar
tidyImportedPlanDataChulwalar <- rawImportedPlanDataChulwalar
tidyImportedIndicatorsChulwalar <- rawImportedIndicatorsChulwalar
```

2. Rename the variables to meaninful labels

```
names(tidyImportedAsIsDataChulwalar) <- c("TotalAsIs", "2008","2009","2010","2011","2
012","2013","2014")
names(tidyImportedPlanDataChulwalar) <- c("TotalPlan", "2008","2009","2010","2011","2
012","2013","2014")
names(tidyImportedIndicatorsChulwalar) <- c("ChangeExportPrices", "2008","2009","2010
","2011","2012","2013","2014")
```

3. Verify dataframe structure. Notice the NA's!

```
str(tidyImportedAsIsDataChulwalar)
```

```
## 'data.frame':    97 obs. of  8 variables:
##  $ TotalAsIs: chr  "Jan" "Feb" "Mar" "Apr" ...
##  $ 2008     : int  2313221 1950131 2346635 2039787 1756964 1458302 1679637 1639670
2882886 2959716 ...
##  $ 2009     : int  2610573 2371327 2743786 2125308 1850073 1836222 1797311 1851968
3271171 2818888 ...
##  $ 2010     : int  2760688 2918333 3227041 1613888 2550157 2317645 1474144 2148521
3898571 3348953 ...
##  $ 2011     : int  3112861 2926663 3294784 2577079 2774068 2378227 2222900 2991787
4151531 3318684 ...
##  $ 2012     : int  3093088 3679308 3433364 2714899 3011767 2726028 2483834 3055655
4200796 4228724 ...
##  $ 2013     : int  4119526 3535744 3560974 3760065 2959933 2787898 2828744 3084113
5107775 4562144 ...
##  $ 2014     : int  4308161 4155378 3924332 3659121 3898758 3313891 3595106 3502426
5619059 5274287 ...
```

```
str(tidyImportedPlanDataChulwalar)
```

```
## 'data.frame':    96 obs. of  8 variables:
##  $ TotalPlan: chr  "Jan" "Feb" "Mar" "Apr" ...
##  $ 2008     : int  2243103 2162705 2720911 2011182 1877757 1819924 1682196 1893171
3325711 2662148 ...
##  $ 2009     : int  2547980 2247049 2731156 2020158 2098038 1927995 1783692 1907705
3124040 3102251 ...
##  $ 2010     : int  2965885 2751170 2906493 2383358 2246893 1992851 2023434 2244997
3257717 3536338 ...
##  $ 2011     : int  3113110 2883766 2957893 2601648 2370949 2339881 2105328 2341623
4086297 3640827 ...
##  $ 2012     : int  3895396 3588151 3787240 3036434 2907891 2707822 2619486 3784557
4987460 4367319 ...
##  $ 2013     : int  3580325 3863212 3606083 3213575 3139128 2998610 2785453 3083654
5143757 4149334 ...
##  $ 2014     : int  4474000 4185565 4278119 3985542 3605973 3515173 3269444 3656112
5637391 5157781 ...
```

```
str(tidyImportedIndicatorsChulwalar)
```

```
## 'data.frame':    194 obs. of  8 variables:
##  $ ChangeExportPrices: chr  "Jan" "Feb" "Mar" "Apr" ...
##  $ 2008              : num  97.4 97.8 98.3 98.1 98.7 98.9 99.5 99.2 99.1 98.9 ...
##  $ 2009              : num  98.3 98.9 98.7 98.8 98.7 99 99 99.2 98.9 98.9 ...
##  $ 2010              : num  99 99.4 99.9 100 99.9 ...
##  $ 2011              : num  101 101 102 102 102 ...
##  $ 2012              : num  103 104 104 104 104 ...
##  $ 2013              : num  104 105 106 105 106 ...
##  $ 2014              : num  NA NA NA NA NA NA NA NA NA NA ...
```

Time Series Data

1. Focus on Total Yearly Exports: Something looks wrong with TotalYearlyExportsAsIs.
2. Discard Year 2014 as there is not Data (NA)

```
tidyImportedAsIsDataChulwalar
```

```
##                     TotalAsIs     2008     2009     2010     2011     2012
## 1                         Jan  2313221  2610573  2760688  3112861  3093088
## 2                         Feb  1950131  2371327  2918333  2926663  3679308
## 3                         Mar  2346635  2743786  3227041  3294784  3433364
## 4                         Apr  2039787  2125308  1613888  2577079  2714899
## 5                         May  1756964  1850073  2550157  2774068  3011767
## 6                         Jun  1458302  1836222  2317645  2378227  2726028
## 7                         Jul  1679637  1797311  1474144  2222900  2483834
## 8                         Aug  1639670  1851968  2148521  2991787  3055655
## 9                         Sep  2882886  3271171  3898571  4151531  4200796
## 10                        Oct  2959716  2818888  3348953  3318684  4228724
## 11                        Nov  2596494  3310776  3135945  4037076  4618540
## 12                        Dec  2656568  3022513  3332886  3429843  3383673
## 13                             NA       NA       NA       NA       NA
## 14                Efak As Is     2008     2009     2010     2011     2012
## 15                        Jan   416589   430055   508177   778643   849409
## 16                        Feb   472565   468187   601115   726254  1021474
## 17                        Mar   466539   648582   775996   943274  1034025
## 18                        Apr   370774   414990   323532   845136   904449
## 19                        May   457741   466329   672011  1030397   986452
## 20                        Jun   384817   465775   589895   829198  1011487
## 21                        Jul   464502   430988   438340   741981   862239
## 22                        Aug   389013   502499   483363   820385  1026357
## 23                        Sep   508370   584983   630064   851428   898892
## 24                        Oct   495598   506877   608942   873895  1079994
## 25                        Nov   529191   593705   688055   996616  1259730
## 26                        Dec   441545   641582   693058   941611   986962
## 27                             NA       NA       NA       NA       NA
## 28                Wuge As Is     2008     2009     2010     2011     2012
## 29                        Jan   414571   462768   525307   507281   545966
## 30                        Feb   344579   393940   515202   564342   632103
```

```
## 31                        Mar   429907   458486   581672   684259   619301
## 32                        Apr   379606   401535   340651   487103   602511
## 33                        Mai   305697   367847   565867   601078   609931
## 34                        Jun   314582   373210   450257   507467   574084
## 35                        Jul   346800   351526   378953   504952   510154
## 36                        Aug   323618   358676   459746   655479   663220
## 37                        Sep   578252   589599   792018   864312   827807
## 38                        Oct   510031   501149   616164   636096   824506
## 39                        Nov   431480   586040   620973   787231   855732
## 40                        Dec   489935   659757   750844   712204   691108
## 41                              NA       NA       NA       NA       NA
## 42          Total Etel As Is    2008     2009     2010     2011     2012
## 43                        Jan  1279668  1583216  1637464  1595267  1519748
## 44                        Feb  1053325  1407388  1676161  1473528  1812897
## 45                        Mar  1367520  1420801  1549560  1469728  1607280
## 46                        Apr  1090725  1141100   813469  1034650  1008022
## 47                        May   873568   919860  1198401   952553  1291983
## 48                        Jun   644479   858876  1140024   819303   940158
## 49                        Jul   772658   910134   551268   802076   945929
## 50                        Aug   806741   843050  1012542  1222812  1235146
## 51                        Sep  1715265  1981563  2335488  2303271  2330334
## 52                        Oct  1795751  1647934  1856264  1591584  2177895
## 53                        Nov  1518288  1857836  1678123  1960675  2306324
## 54                        Dec  1601324  1615091  1699063  1713991  1618147
## 55                              NA       NA       NA       NA       NA
## 56         Blue Etel As Is      2008     2009     2010     2011     2012
## 57                        Jan   425892   407424   369783   308893   285207
## 58                        Feb   316631   287654   345144   282106   450874
## 59                        Mar   353512   305158   322695   347124   360034
## 60                        Apr   278711   255687   223841   261498   252674
## 61                        May   212940   200068   239441   217606   247734
## 62                        Jun   187849   210118   240316   208258   221676
## 63                        Jul   206285   211668   138604   174878   216918
## 64                        Aug   195810   198472   231179   247714   254993
## 65                        Sep   448733   361703   329090   312012   299658
## 66                        Oct   403327   366410   368584   331926   457595
## 67                        Nov   306171   350196   320947   389858   388917
## 68                        Dec   345955   351651   373302   299115   303450
## 69                              NA       NA       NA       NA       NA
## 70          Red Etel As Is      2008     2009     2010     2011     2012
## 71                        Jan   853776  1175792  1267682  1286374  1234541
## 72                        Feb   736694  1119734  1331017  1191422  1362023
## 73                        Mar  1014008  1115643  1226866  1122604  1247246
## 74                        Apr   812014   885413   589628   773151   755347
## 75                        May   660628   719792   958960   734947  1044249
## 76                        Jun   456630   648758   899709   611045   718482
## 77                        Jul   566373   698466   412664   627198   729011
## 78                        Aug   610931   644578   781363   975098   980154
## 79                        Sep  1266532  1619860  2006398  1991259  2030676
## 80                        Oct  1392424  1281524  1487680  1259658  1720301
```

```
## 81                          Nov  1212117  1507640  1357176  1570817  1917408
## 82                          Dec  1255369  1263440  1325761  1414876  1314697
## 83                                    NA       NA       NA       NA       NA
## 84 Total yearly Exports As Is      2008     2009     2010     2011     2012
## 85                          Jan 26280011 29609916 32726772 37215503 40629676
## 86                          Feb 26280011 29609916 32726772 37215503 40629676
## 87                          Mar 26280011 29609916 32726772 37215503 40629676
## 88                          Apr 26280011 29609916 32726772 37215503 40629676
## 89                          May 26280011 29609916 32726772 37215503 40629676
## 90                          Jun 26280011 29609916 32726772 37215503 40629676
## 91                          Jul 26280011 29609916 32726772 37215503 40629676
## 92                          Aug 26280011 29609916 32726772 37215503 40629676
## 93                          Sep 26280011 29609916 32726772 37215503 40629676
## 94                          Oct 26280011 29609916 32726772 37215503 40629676
## 95                          Nov 26280011 29609916 32726772 37215503 40629676
## 96                          Dez 26280011 29609916 32726772 37215503 40629676
## 97                                    NA       NA       NA       NA       NA
##        2013    2014
## 1   4119526 4308161
## 2   3535744 4155378
## 3   3560974 3924332
## 4   3760065 3659121
## 5   2959933 3898758
## 6   2787898 3313891
## 7   2828744 3595106
## 8   3084113 3502426
## 9   5107775 5619059
## 10  4562144 5274287
## 11  4729313 4841693
## 12  4372181 4664854
## 13       NA      NA
## 14     2013      NA
## 15  1065097      NA
## 16   952195      NA
## 17  1062892      NA
## 18  1057988      NA
## 19  1127932      NA
## 20   933365      NA
## 21  1069867      NA
## 22  1020078      NA
## 23  1049970      NA
## 24  1197452      NA
## 25  1283970      NA
## 26  1280835      NA
## 27       NA      NA
## 28     2013      NA
## 29   752685      NA
## 30   708242      NA
## 31   719168      NA
## 32   787368      NA
```

```
## 33   574721      NA
## 34   643629      NA
## 35   628135      NA
## 36   718542      NA
## 37   923583      NA
## 38   934234      NA
## 39   886772      NA
## 40   948935      NA
## 41       NA      NA
## 42     2013      NA
## 43  2109497      NA
## 44  1738197      NA
## 45  1633944      NA
## 46  1745092      NA
## 47  1039449      NA
## 48  1054201      NA
## 49  1003166      NA
## 50  1154675      NA
## 51  3000929      NA
## 52  2305605      NA
## 53  2284672      NA
## 54  2062160      NA
## 55       NA      NA
## 56     2013      NA
## 57   387497      NA
## 58   349013      NA
## 59   334274      NA
## 60   325052      NA
## 61   255416      NA
## 62   237019      NA
## 63   239047      NA
## 64   358552      NA
## 65   359703      NA
## 66   427681      NA
## 67   434561      NA
## 68   348558      NA
## 69       NA      NA
## 70     2013      NA
## 71  1722000      NA
## 72  1389184      NA
## 73  1299670      NA
## 74  1420039      NA
## 75   784033      NA
## 76   817182      NA
## 77   764120      NA
## 78   796123      NA
## 79  2641226      NA
## 80  1877924      NA
## 81  1850111      NA
## 82  1713603      NA
```

```
## 83       NA      NA
## 84     2013      NA
## 85 45408410      NA
## 86 45408410      NA
## 87 45408410      NA
## 88 45408410      NA
## 89 45408410      NA
## 90 45408410      NA
## 91 45408410      NA
## 92 45408410      NA
## 93 45408410      NA
## 94 45408410      NA
## 95 45408410      NA
## 96 45408410      NA
## 97       NA      NA
```

```
TotalYearlyExports <- tidyImportedAsIsDataChulwalar[84:97,]
TotalYearlyExports <- TotalYearlyExports[c(1:7)]
TotalYearlyExports
```

```
##                       TotalAsIs     2008     2009     2010     2011     2012
## 84 Total yearly Exports As Is     2008     2009     2010     2011     2012
## 85                          Jan 26280011 29609916 32726772 37215503 40629676
## 86                          Feb 26280011 29609916 32726772 37215503 40629676
## 87                          Mar 26280011 29609916 32726772 37215503 40629676
## 88                          Apr 26280011 29609916 32726772 37215503 40629676
## 89                          May 26280011 29609916 32726772 37215503 40629676
## 90                          Jun 26280011 29609916 32726772 37215503 40629676
## 91                          Jul 26280011 29609916 32726772 37215503 40629676
## 92                          Aug 26280011 29609916 32726772 37215503 40629676
## 93                          Sep 26280011 29609916 32726772 37215503 40629676
## 94                          Oct 26280011 29609916 32726772 37215503 40629676
## 95                          Nov 26280011 29609916 32726772 37215503 40629676
## 96                          Dez 26280011 29609916 32726772 37215503 40629676
## 97                                    NA       NA       NA       NA       NA
##        2013
## 84     2013
## 85 45408410
## 86 45408410
## 87 45408410
## 88 45408410
## 89 45408410
## 90 45408410
## 91 45408410
## 92 45408410
## 93 45408410
## 94 45408410
## 95 45408410
## 96 45408410
## 97       NA
```

```
TotalYearlyExports_vector <- c(26280011, 29609916, 32726772, 37215503, 40629676, 4540
8410)
TotalYearlyExports_vector
```

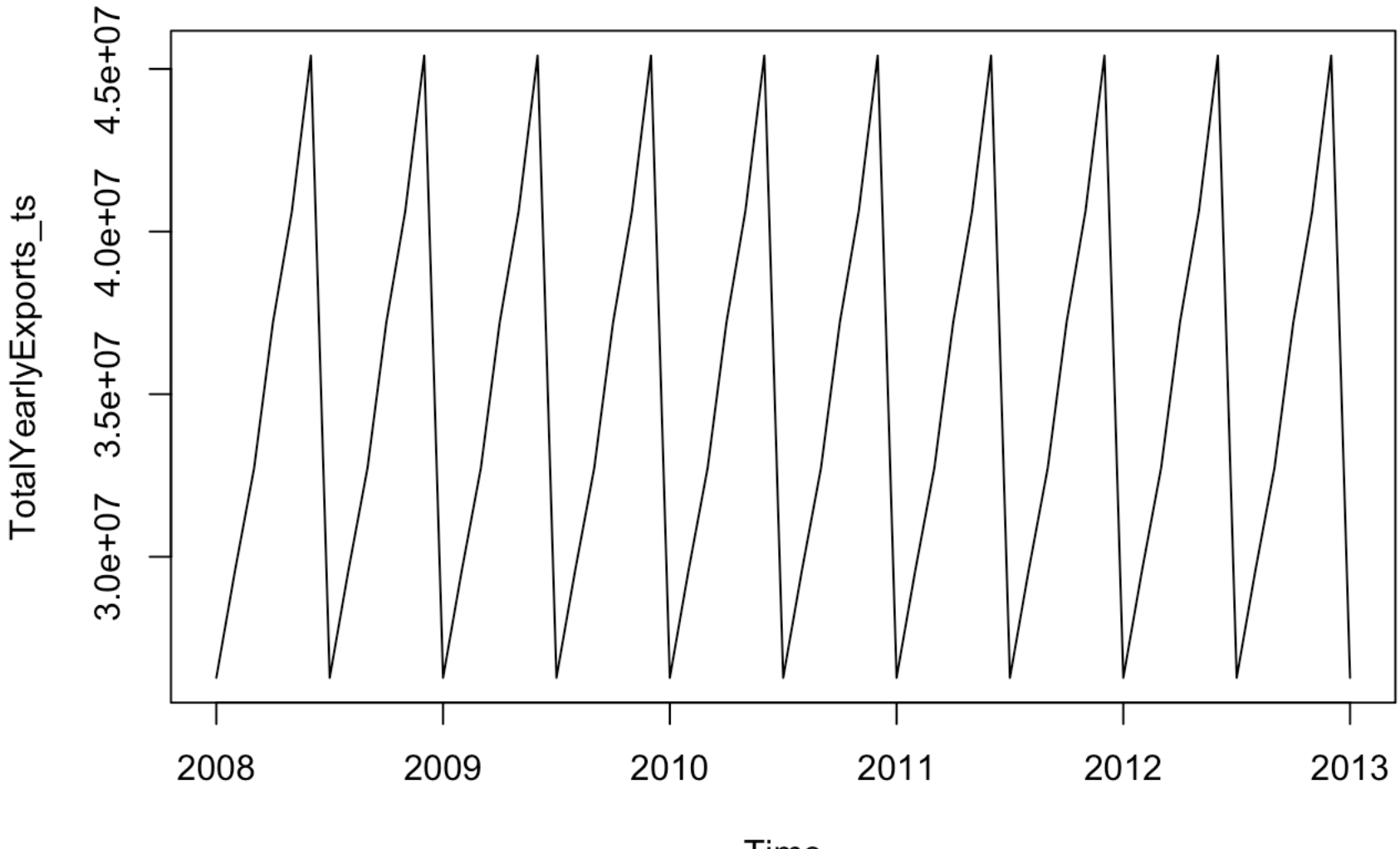
```
## [1] 26280011 29609916 32726772 37215503 40629676 45408410
```

2. Create a timeseries datatype

install.packages("fpp", repos="http://cran.us.r-project.org (http://cran.us.r-project.org/)",dependencies=TRUE)

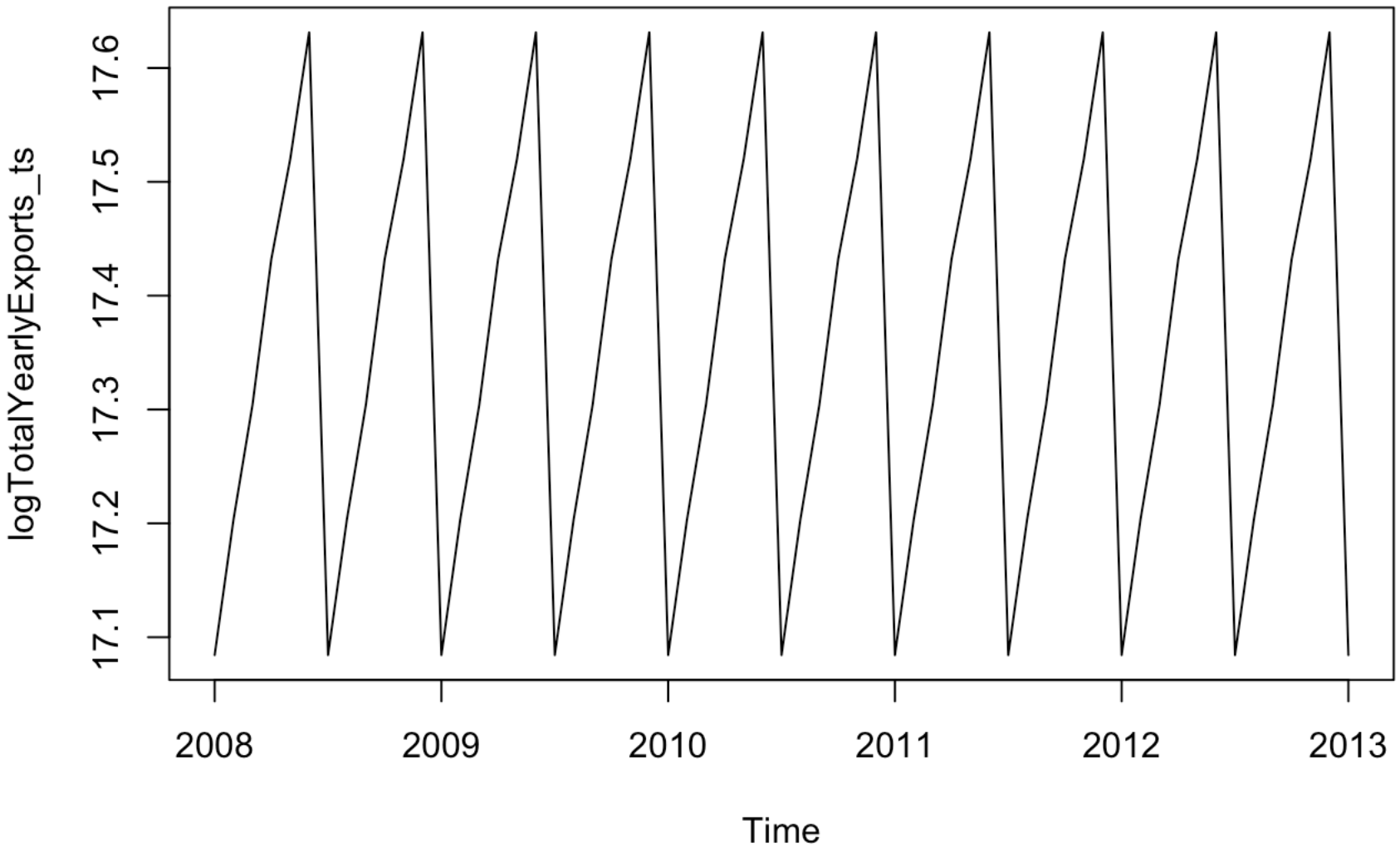
library(fpp)

```
TotalYearlyExports_ts <- ts(TotalYearlyExports_vector, start=c(2008), end=c(2013), fr
equency=12)
plot.ts(TotalYearlyExports_ts)
```



3. We can notice the fluctuations over time are constant showing a definite seasonality. As a secondary reference we can transform the time series by calculating the natural log of the original data:

```
logTotalYearlyExports_ts <- log(TotalYearlyExports_ts)
plot.ts(logTotalYearlyExports_ts)
```

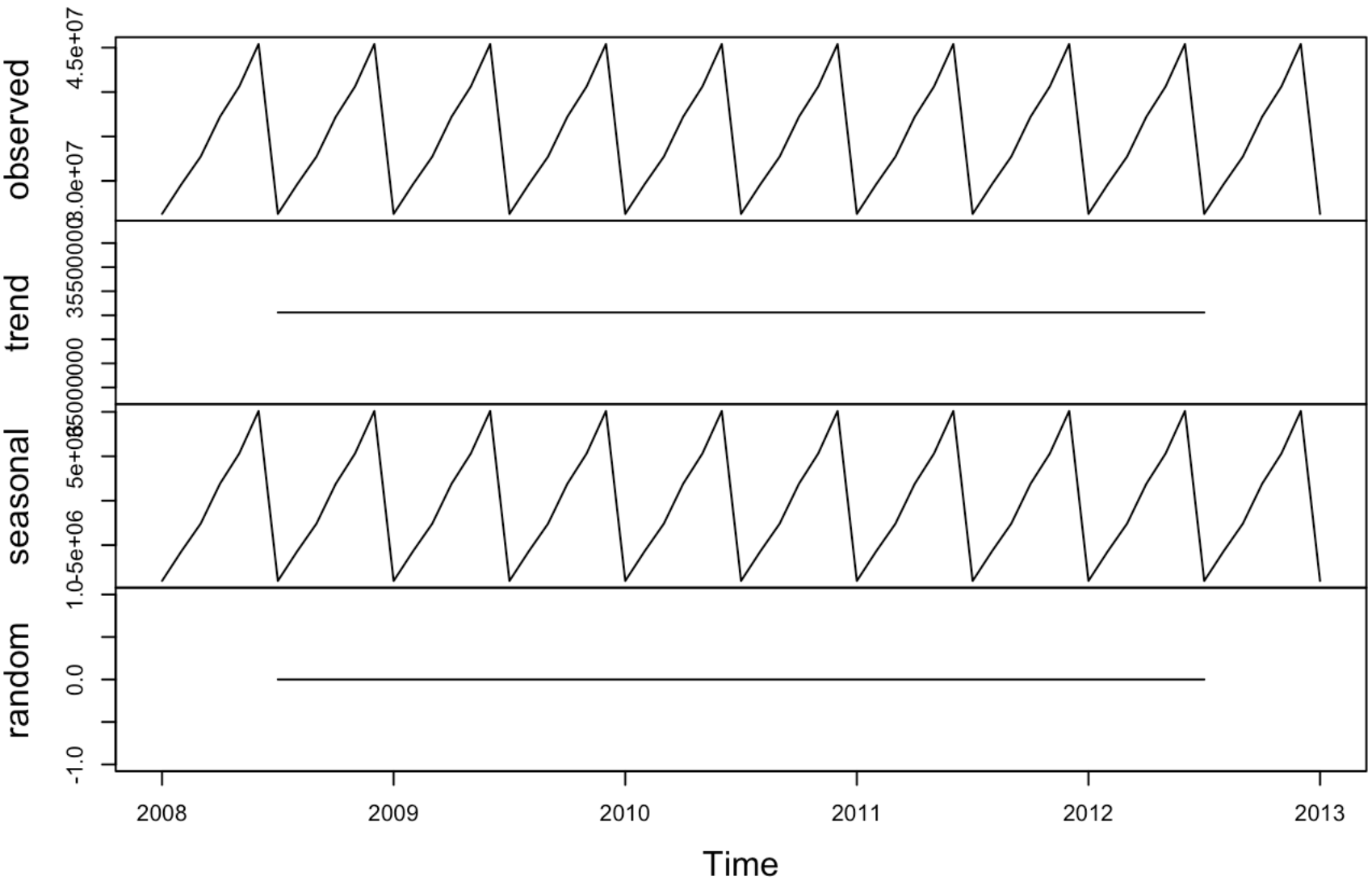


Decomposition

“Decomposing a time series means separating it into its constituent components, which are usually a trend component and an irregular component, and if it is a seasonal time series, a seasonal component.” https://a-little-book-of-r-for-time-series.readthedocs.io/en/latest/src/timeseries.html (https://a-little-book-of-r-for-time-series.readthedocs.io/en/latest/src/timeseries.html)

1. The next steps will be to use a basic decomposition to observe the trend cycle and seasonality of the data

```
decompose_TotalYearlyExports_ts <- decompose(TotalYearlyExports_ts)
plot(decompose_TotalYearlyExports_ts)
```



2. Observe the estimated values: The largest seasonal factor seems to occur aproximately every six months (June and Dec 10096695)

```
decompose_TotalYearlyExports_ts$seasonal
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 2008 -9031704 -5701799 -2584943  1903788  5317961 10096695 -9031704
## 2009 -9031704 -5701799 -2584943  1903788  5317961 10096695 -9031704
## 2010 -9031704 -5701799 -2584943  1903788  5317961 10096695 -9031704
## 2011 -9031704 -5701799 -2584943  1903788  5317961 10096695 -9031704
## 2012 -9031704 -5701799 -2584943  1903788  5317961 10096695 -9031704
## 2013 -9031704
##           Aug      Sep      Oct      Nov      Dec
## 2008 -5701799 -2584943  1903788  5317961 10096695
## 2009 -5701799 -2584943  1903788  5317961 10096695
## 2010 -5701799 -2584943  1903788  5317961 10096695
## 2011 -5701799 -2584943  1903788  5317961 10096695
## 2012 -5701799 -2584943  1903788  5317961 10096695
## 2013
```

Correlation of Imported Indicators

See if the results above support the imported indicators: Indicator chosen Chage in Export Prices.

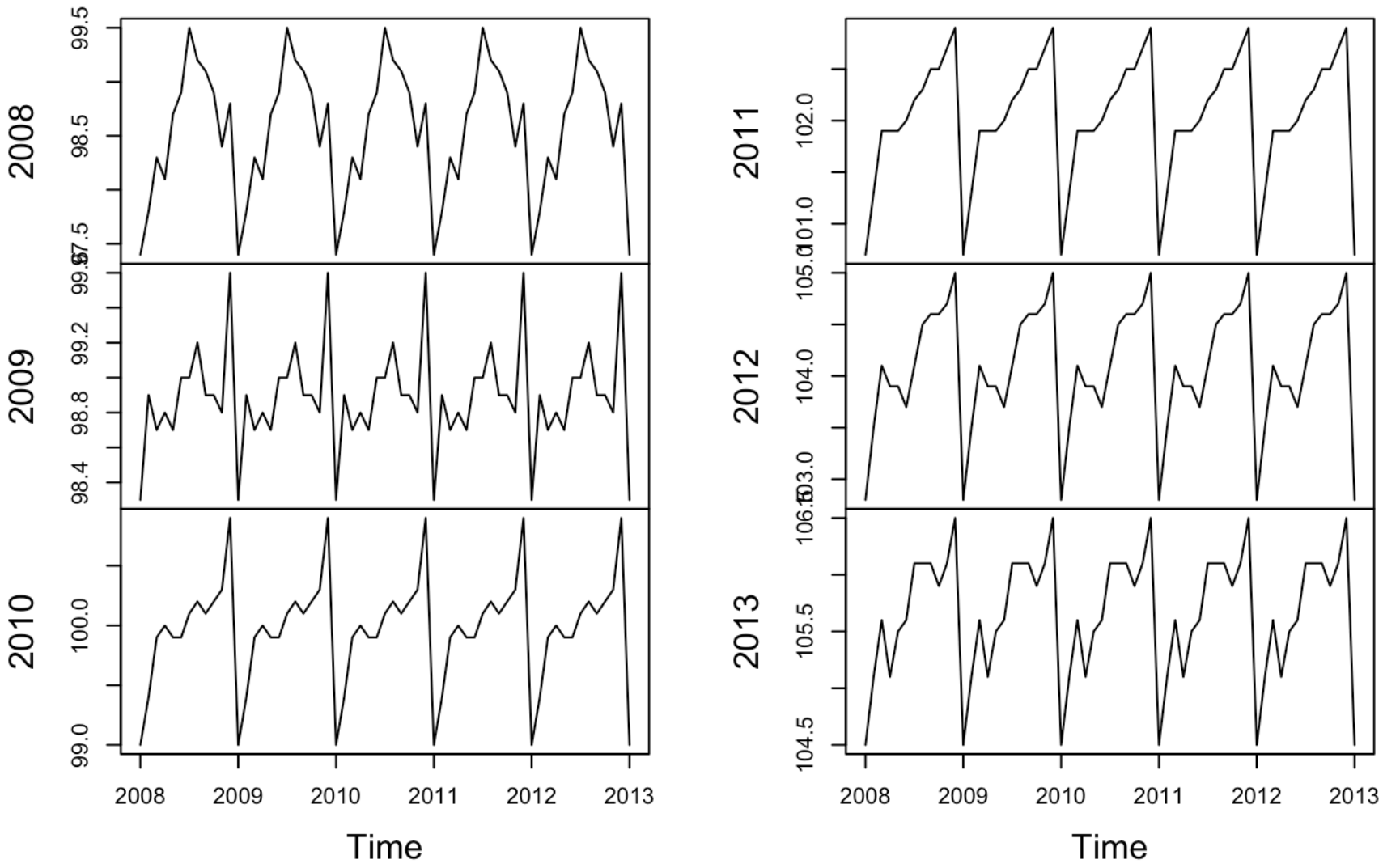
```
ChangeExportPrices <- tidyImportedIndicatorsChulwalar[1:12,]
ChangeExportPrices <- ChangeExportPrices[c(2:7)]
ChangeExportPrices
```

```
##    2008 2009  2010  2011  2012  2013
## 1  97.4 98.3  99.0 100.7 102.8 104.5
## 2  97.8 98.9  99.4 101.3 103.5 105.1
## 3  98.3 98.7  99.9 101.9 104.1 105.6
## 4  98.1 98.8 100.0 101.9 103.9 105.1
## 5  98.7 98.7  99.9 101.9 103.9 105.5
## 6  98.9 99.0  99.9 102.0 103.7 105.6
## 7  99.5 99.0 100.1 102.2 104.1 106.1
## 8  99.2 99.2 100.2 102.3 104.5 106.1
## 9  99.1 98.9 100.1 102.5 104.6 106.1
## 10 98.9 98.9 100.2 102.5 104.6 105.9
## 11 98.4 98.8 100.3 102.7 104.7 106.1
## 12 98.8 99.6 100.9 102.9 105.0 106.5
```

```
ChangeExportPrices_ts <- ts(ChangeExportPrices, start=c(2008), end=c(2013), frequency
=12)
```

We can observe that peaks also occurr in June for the year of 2008 and December for all other months.

```
plot.ts(ChangeExportPrices_ts)
```



Proceed to Forecast

Install required pacakges:

```
install.packages("forecast", repos="http://cran.us.r-project.org",dependencies=TRUE)
```

```
##
## The downloaded binary packages are in
##  /var/folders/dg/wtvtq1m96q9cc_33_fpwv9d40000gn/T//RtmpjdsDn9/downloaded_packages
```

```
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 3.2.5
```

```
## Loading required package: zoo
```

```
## Warning: package 'zoo' was built under R version 3.2.5
```

```
##
## Attaching package: 'zoo'
```

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

```
## Loading required package: timeDate
```

```
## This is forecast 7.1
```

ARIMA forecasting model

---

1. Build the ARIMA forecasting model: the arima function takes the time series data as an input, and parameters (p, d, q), defined as follows: p: The number of autoregressive terms d: The number of nonseasonal differences needed for stationarity q: The number of lagged forecast errors

   Some combinations are:

   (1,0,0): This series is generally used when the data is highly auto-correlated. Here, we predict the current value using its immediate preceding value. Usually, the current value will be predicted by multiplying the previous value with a multiplicative factor plus a constant.

   (2,0,0): In this case, the current value is predicted based on the preceding two values.

   (0,1,0): This is also called a random walk model, where there is neither seasonality nor can the previous value predict the current value.

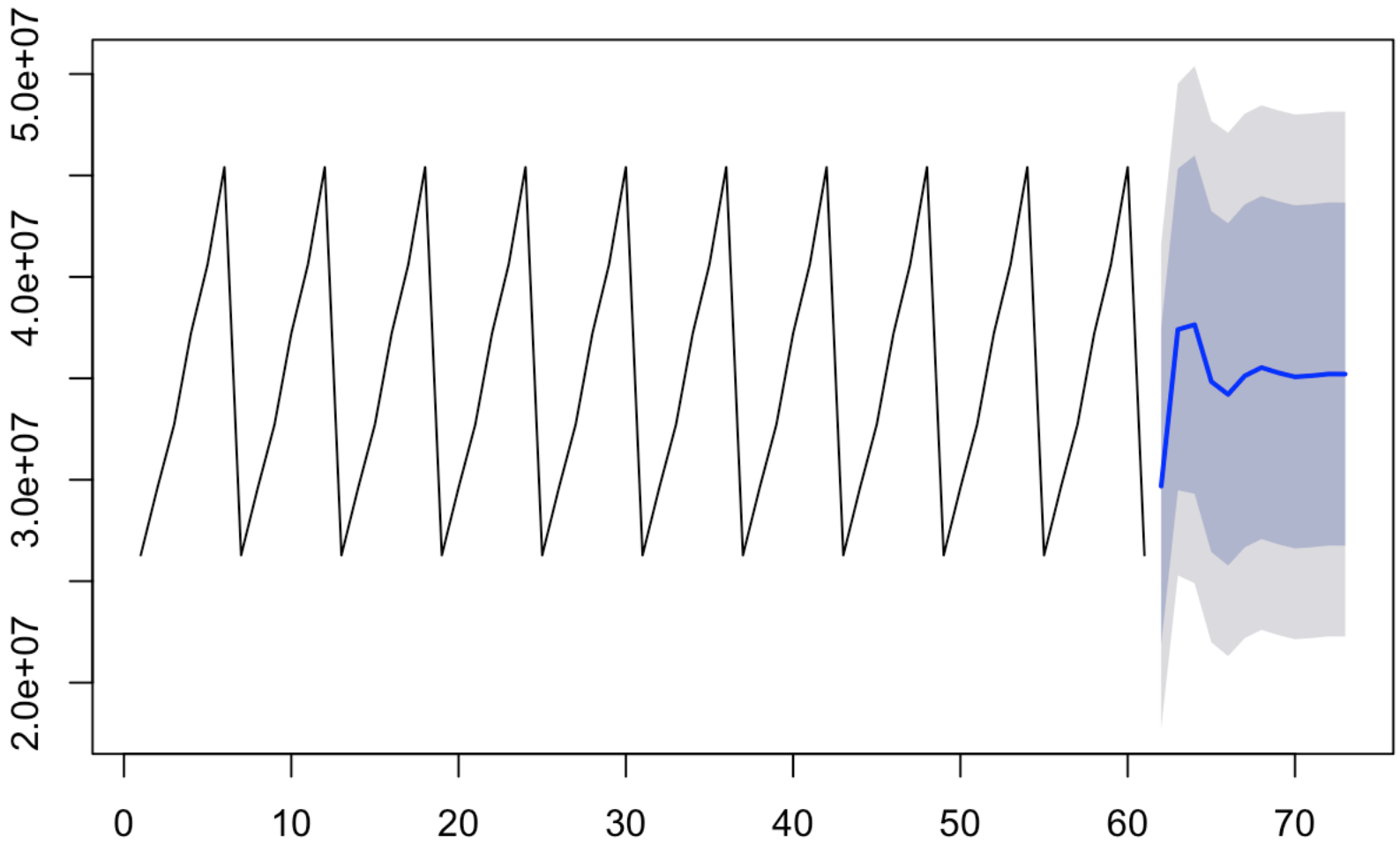
   (1,1,0): This case is used when the random walk model has a drift that can be defined by the previous value.

   (0,1,1): This method can be used when there is no seasonality or trend in the data. This is also called the simple exponential smoothening method. In this case, the error of the random walk is offset by the smoothening of the previous value.

```
a_model=arima(as.matrix(TotalYearlyExports_ts), order=c(2,0,0))
```

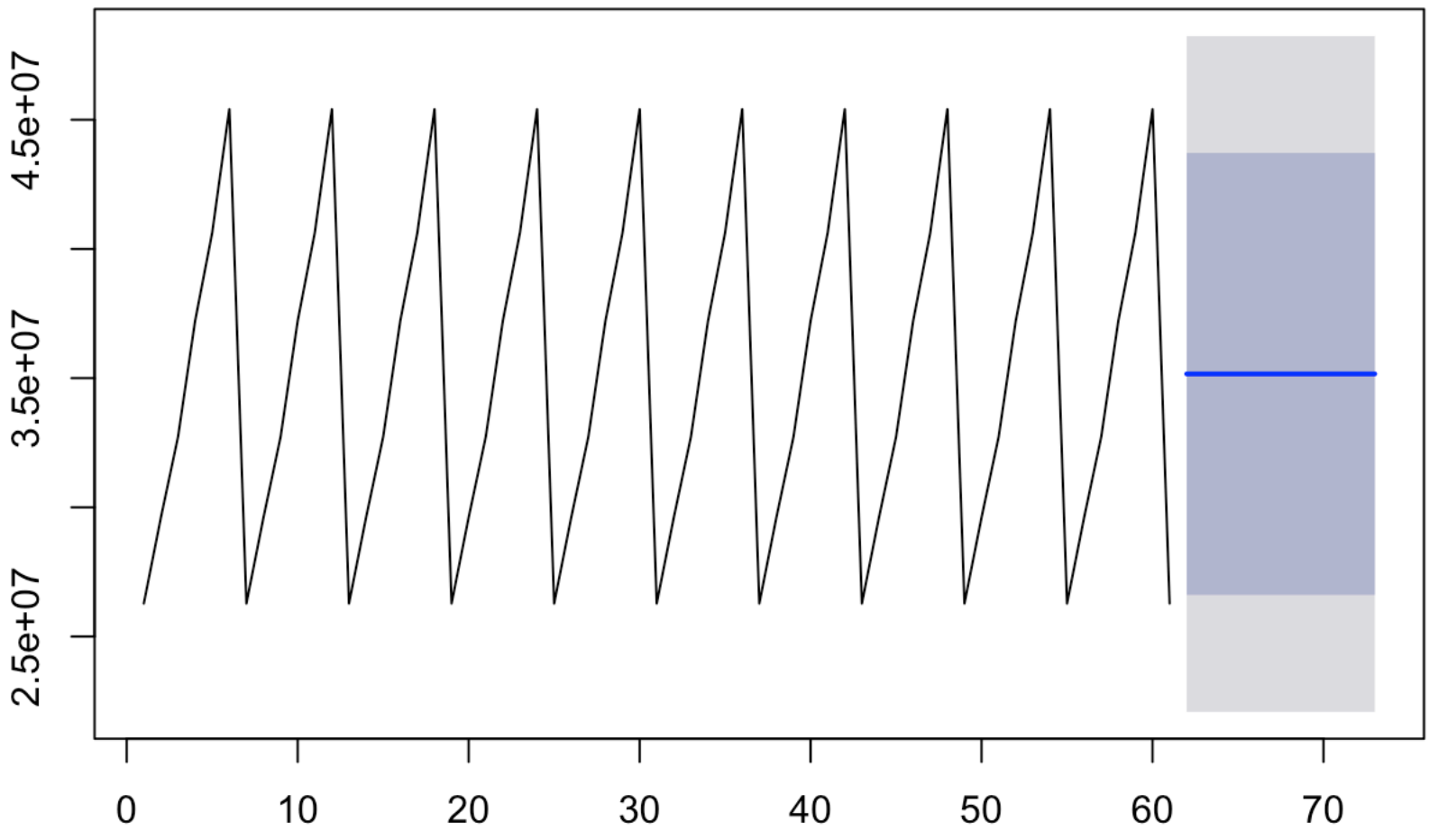
2. Forecast: pass the arima model itself as a parameter to the forecast function, we also need to pass the number of future values that has to be predicted.

```
aforecast=forecast.Arima(a_model,h=12)
plot(aforecast)
```



3. Conclusions: We chose model 2,0,0 as seasonality is highly evident in the data and think this model is more apropiate than others. For example when we try to see how simple exponential smoothening model will look we notice forecasted numbers appear to be less accurate: [We keep tweaking the parameters to improve accuracy].

```
a_model=arima(as.matrix(TotalYearlyExports_ts), order=c(0,1,1))
aforecast=forecast.Arima(a_model,h=12)
plot(aforecast)
```



Holt-Winters method

1. The HoltWinters function requires a given time series data, and the following parameters:

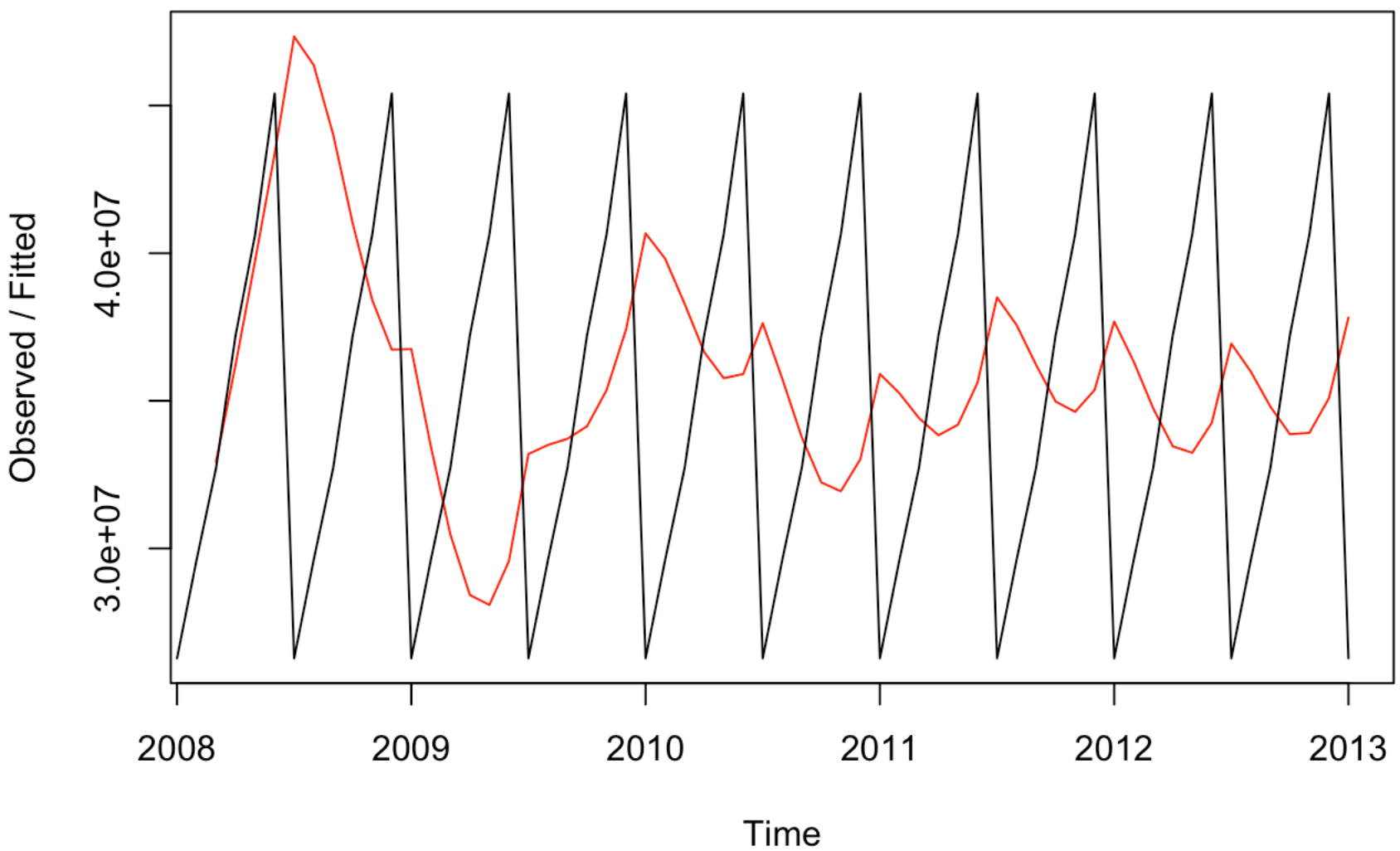
   alpha: The parameter of the Holt-Winters filters

   beta: This is used for the trend component; when set to false, the function will do exponential smoothening

   gamma: This is used for the seasonal component; when set to false, the nonseasonal component is fitted

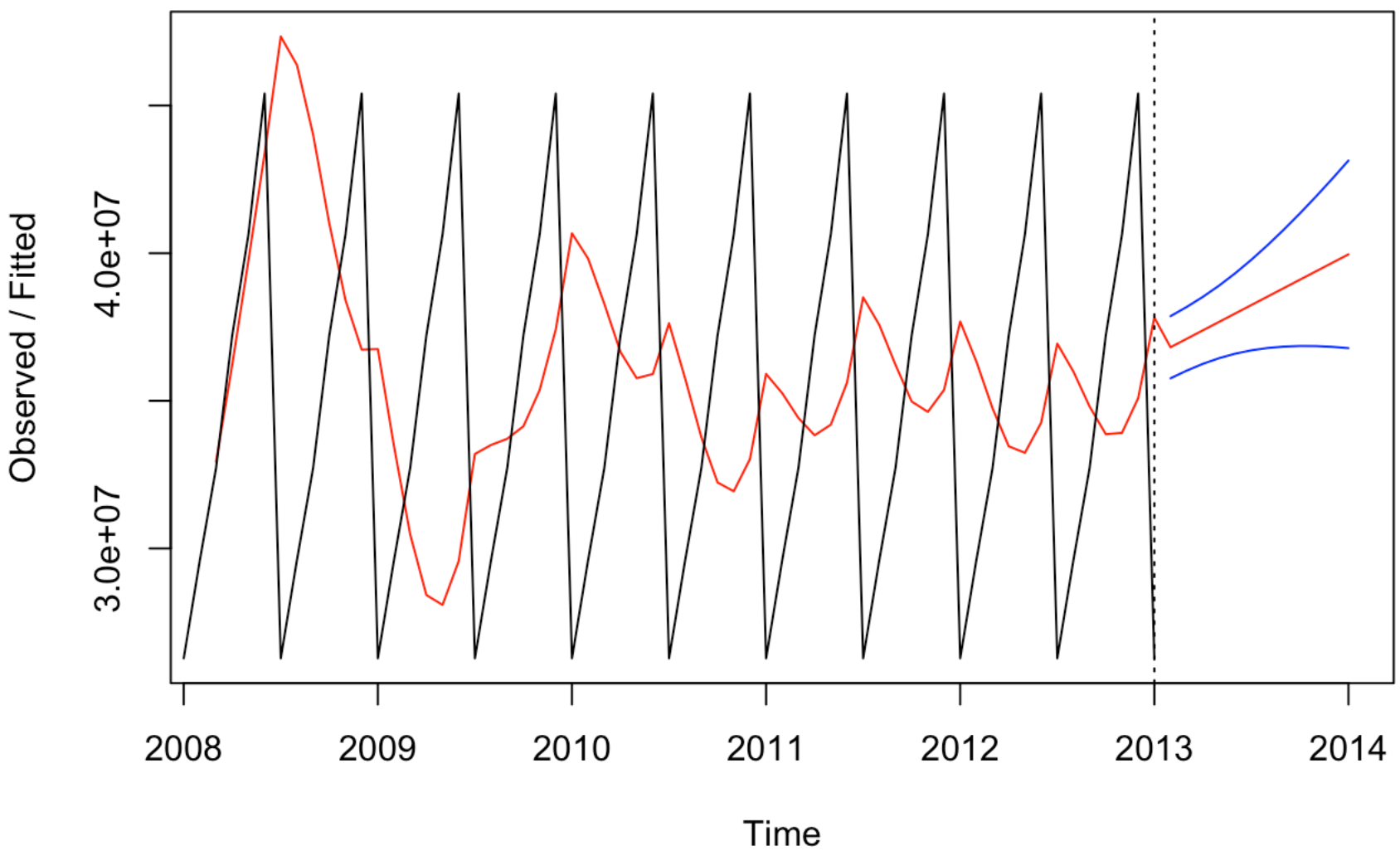
2. Here again we buil the model we think is more appropiate, in which case setting beta=TRUE to avoid doing exponential smoothering.

```
h_model=HoltWinters(TotalYearlyExports_ts, beta=TRUE, gamma=FALSE)
plot(h_model)
```



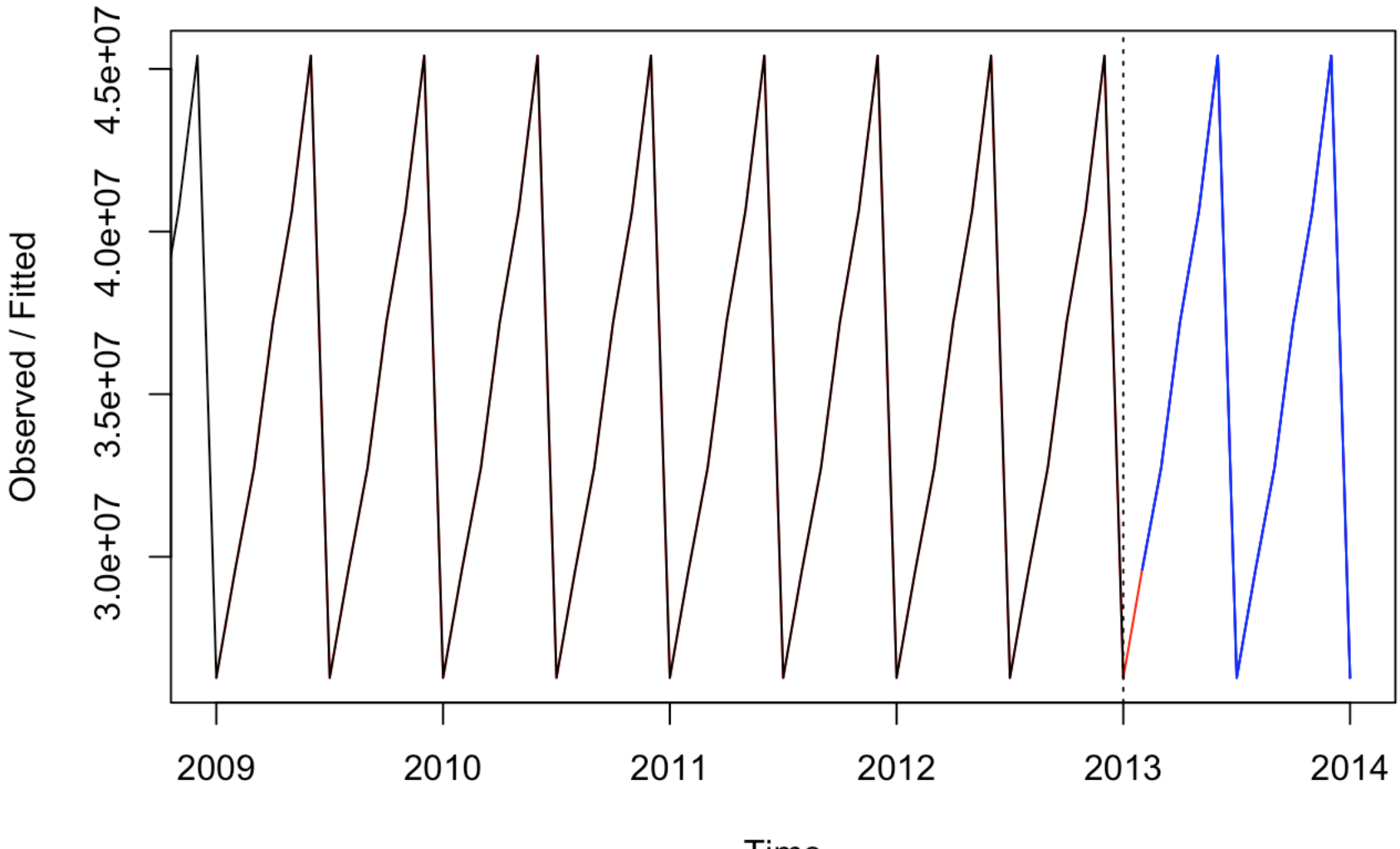
3. Forecast: We pass the model that ww built to the predict function. Other parameters are: the period (for which the prediction has to be made), prediction.interval (when set to true, the lower and upper bounds of the corresponding prediction intervals are computed), and level (which is the confidence interval for the prediction).

```
forecast <- predict(h_model, n.ahead = 12, prediction.interval = T, level = 0.1)
plot(h_model, forecast)
```



4. Conclusion: Our chosen model looks off. It looks like after all we might need to adjust the gamma value. We decided to try one more time and notice this second attempt is more in line with historical data [We keep tweaking the parameters to improve accuracy]:

```
h_model=HoltWinters(TotalYearlyExports_ts, beta=TRUE, gamma=0.1)
forecast <- predict(h_model, n.ahead = 12, prediction.interval = T, level = 0.1)
plot(h_model, forecast)
```



5. See actual forcasted values:

```
hforecast <- as.data.frame(forecast)
forecast_exports <- hforecast$fit
forecast_exports <- as.data.frame(forecast_exports)
head(forecast_exports, 10)
```

```
##    forecast_exports
## 1          29609916
## 2          32726772
## 3          37215503
## 4          40629676
## 5          45408410
## 6          26280011
## 7          29609916
## 8          32726772
## 9          37215503
## 10         40629676
```

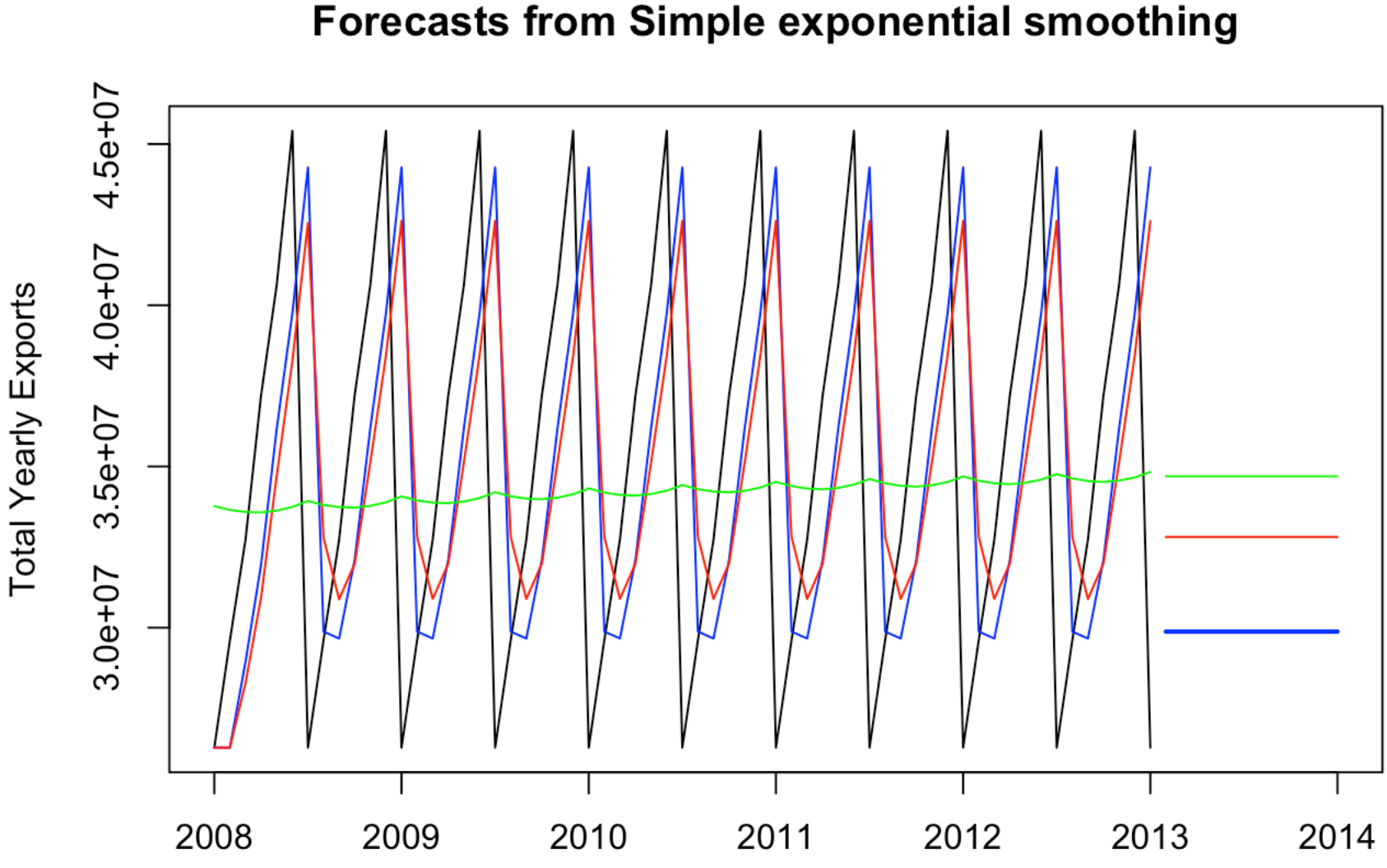
Simple Exponential Smothing

“Forecasts produced using simple exponential smoothing methods are weighted averages of past observations, with the weights decaying exponentially as the observations get older”.

https://www.otexts.org/fpp/7 (https://www.otexts.org/fpp/7)

1. First: try the model with an alpha smoothing parameter of 0.8
2. Second: try the model with an alpha smoothing parameter of 0.6
3. Third: all defaults

```
fit1 <- ses(TotalYearlyExports_ts, alpha=0.8, beta=0.2, initial="simple", h=12)
fit2 <- ses(TotalYearlyExports_ts, alpha=0.6, beta=0.2, initial="simple", h=12 )
fit3 <- ses(TotalYearlyExports_ts, h=12 )
plot(fit1, plot.conf=FALSE, ylab="Total Yearly Exports")
lines(fitted(fit1), col="blue")
lines(fitted(fit2), col="red")
lines(fitted(fit3), col="green")
lines(fit1$mean, col="blue")
lines(fit2$mean, col="red")
lines(fit3$mean, col="green")
```



4. Conclusion: We noticed the blue line is more accurate to our historical data (when alpha=0.8) meaning

is better when more weight is giving to past values.

Here the actual predicted values: Defaults (green) gives us more optimistics values.

```
fit1$mean
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 2013          29879066 29879066 29879066 29879066 29879066 29879066
## 2014 29879066
##           Aug      Sep      Oct      Nov      Dec
## 2013 29879066 29879066 29879066 29879066 29879066
## 2014
```

```
fit2$mean
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 2013          32814564 32814564 32814564 32814564 32814564 32814564
## 2014 32814564
##           Aug      Sep      Oct      Nov      Dec
## 2013 32814564 32814564 32814564 32814564 32814564
## 2014
```

```
fit3$mean
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 2013          34694426 34694426 34694426 34694426 34694426 34694426
## 2014 34694426
##           Aug      Sep      Oct      Nov      Dec
## 2013 34694426 34694426 34694426 34694426 34694426
## 2014
```

Holt's Linear Trend Method

This method allows for forecasting data with a trend. The trend component observed in the historical data is pretty flat so we decided to try this model at the very last. It also uses alpha as smoothing parameter for the level and beta as the smoothing parameter for the trend. It can be either additive or exponential.

1. First: try additive model with a smoothing parameter of 0.8
2. Second: try an exponential model with a smoothing parameter of 0.6
3. Third: all defaults

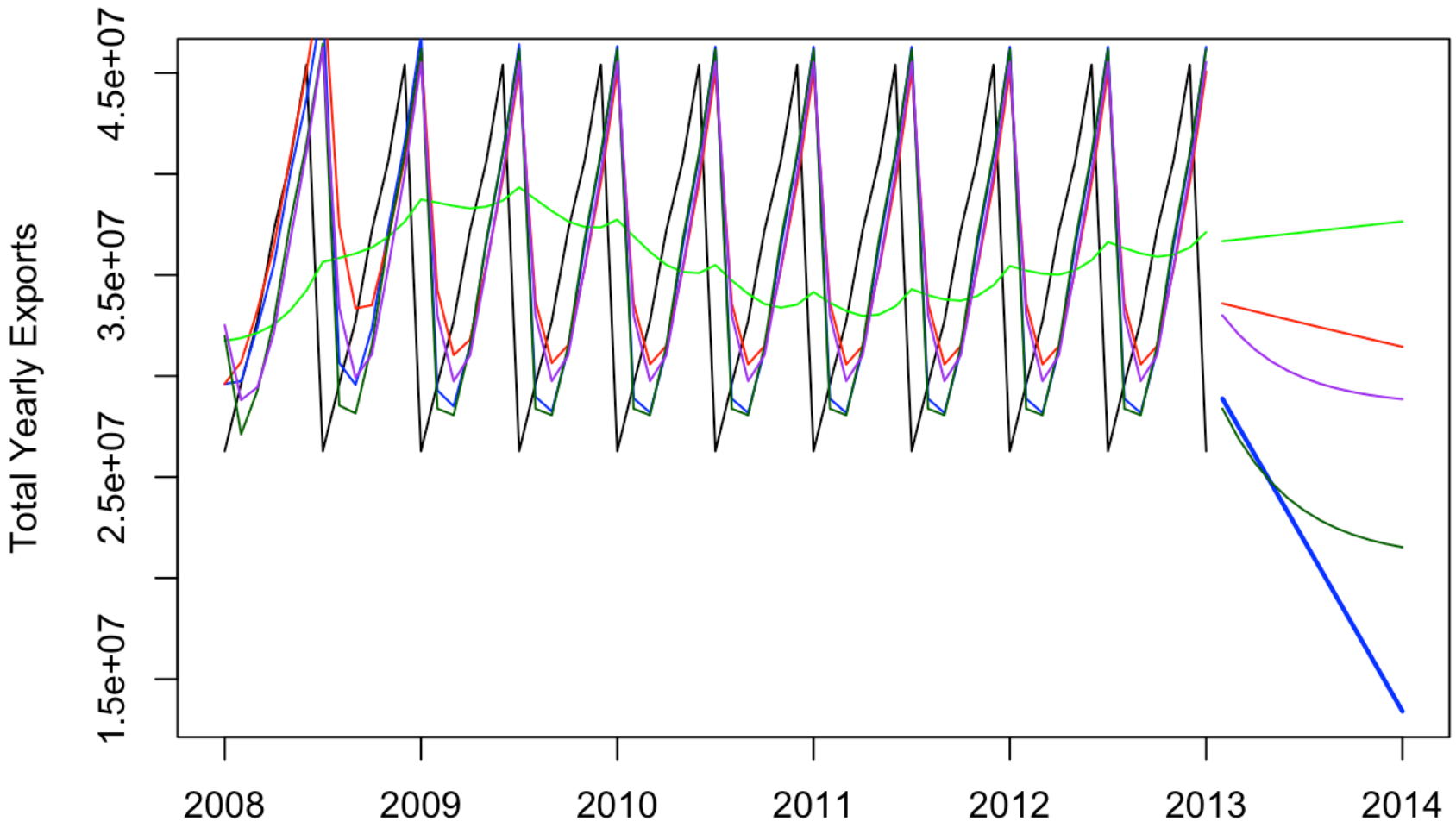
```
fit1 <- holt(TotalYearlyExports_ts, alpha=.8, beta=0.2, initial="simple", h=12)
fit2 <- holt(TotalYearlyExports_ts, alpha=.6, beta=0.2, initial="simple", exponential
=TRUE, h=12 )
fit3 <- holt(TotalYearlyExports_ts, h=12 )
plot(fit1, plot.conf=FALSE, ylab="Total Yearly Exports")
lines(fitted(fit1), col="blue")
lines(fitted(fit2), col="red")
lines(fitted(fit3), col="green")
lines(fit1$mean, col="blue")
lines(fit2$mean, col="red")
lines(fit3$mean, col="green")
fit4 <- holt(TotalYearlyExports_ts, alpha=.8, beta=0.2, damped=TRUE, initial="simple"
, h=12)
```

```
## Warning in holt(TotalYearlyExports_ts, alpha = 0.8, beta = 0.2, damped =
## TRUE, : Damped Holt's method requires optimal initialization
```

```
lines(fitted(fit4), col="darkgreen")
lines(fit4$mean, col="darkgreen")
fit5 <- holt(TotalYearlyExports_ts, alpha=.6, beta=0.2, damped=TRUE, initial="simple"
, exponential=TRUE, h=12 )
```

```
## Warning in holt(TotalYearlyExports_ts, alpha = 0.6, beta = 0.2, damped =
## TRUE, : Damped Holt's method requires optimal initialization
```

```
lines(fitted(fit5), col="purple")
lines(fit5$mean, col="purple")
```



4. Conclusion: As we could observed (and had expected), there is no trend really defined, or one that could be forecast using this model. Since the historical data shows seasonality component rather than a trend.

Here the actual predicted values: Notice that both, additive and exponential models propose a negative trend.

```
fit1$mean
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 2013          28876333 27469987 26063641 24657295 23250950 21844604
## 2014 13406529
##           Aug      Sep      Oct      Nov      Dec
## 2013 20438258 19031912 17625567 16219221 14812875
## 2014
```

```
fit2$mean
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 2013          33593471 33392594 33192918 32994437 32797142 32601027
## 2014 31448720
##           Aug      Sep      Oct      Nov      Dec
## 2013 32406085 32212309 32019691 31828225 31637904
## 2014
```

```
fit3$mean
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 2013          36671249 36760439 36849629 36938818 37028008 37117197
## 2014 37652334
##           Aug      Sep      Oct      Nov      Dec
## 2013 37206387 37295576 37384766 37473955 37563145
## 2014
```

```
fit4$mean
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 2013          28381873 26882041 25682177 24722285 23954371 23340040
## 2014 21526889
##           Aug      Sep      Oct      Nov      Dec
## 2013 22848575 22455404 22140866 21889236 21687932
## 2014
```

```
fit5$mean
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 2013          33002671 32046772 31302026 30718710 30259894 29897780
## 2014 28854100
##           Aug      Sep      Oct      Nov      Dec
## 2013 29611211 29383935 29203370 29059718 28945305
## 2014
```

Damp Trend Method

Often the single-most accurate forecasting method for seasonal data!

Damped Linear Method It can be additive or exponentially (less conservative), and includes a damp paremeter.

Here is the code we used for the damped version:

fit4 <- holt(TotalYearlyExports_ts, alpha=.8, beta=0.2, damped=TRUE, initial="simple", h=12) lines(fitted(fit4), col="darkgreen") fit4$meanlines(fit3$mean, col="green")

fit5 <- holt(TotalYearlyExports_ts, alpha=.6, beta=0.2, damped=TRUE, initial="simple", exponential=TRUE, h=12) lines(fitted(fit5), col="purple") lines(fit5$mean, col = " purple ")fit5$mean

Conclusion: Fit5 model is slightly better but not quite convincing when compared against historical data.

Accuracy

1. Compare the different the models:

```
fit1$model
```

```
## Holt's method
##
## Call:
##  holt(x = TotalYearlyExports_ts, h = 12, initial = "simple", alpha = 0.8,
##
##  Call:
##      beta = 0.2)
##
##   Smoothing parameters:
##     alpha = 0.8
##     beta  = 0.2
##
##   Initial states:
##     l = 26280011
##     b = 3329905
##
##   sigma:  9014603
```

```
fit2$model
```

```
## Holt's method with exponential trend
##
## Call:
##  holt(x = TotalYearlyExports_ts, h = 12, initial = "simple", exponential = TRUE,
##
##  Call:
##      alpha = 0.6, beta = 0.2)
##
##   Smoothing parameters:
##     alpha = 0.6
##     beta  = 0.2
##
##   Initial states:
##     l = 26280011
##     b = 1.1267
##
##   sigma:  0.2091
```

```
fit3$model
```

```
## Holt's method
##
## Call:
##  holt(x = TotalYearlyExports_ts, h = 12)
##
##   Smoothing parameters:
##     alpha = 0.0497
##     beta  = 0.0205
##
##   Initial states:
##     l = 31216068.5903
##     b = 523612.6618
##
##   sigma:  6985125
##
##      AIC     AICc      BIC
## 2181.397 2182.111 2189.841
```

```
fit4$model
```

```
## Damped Holt's method
##
## Call:
##  holt(x = TotalYearlyExports_ts, h = 12, damped = TRUE, initial = "simple",
##
##  Call:
##      alpha = 0.8, beta = 0.2)
##
##   Smoothing parameters:
##     alpha = 0.8
##     beta  = 0.2
##     phi   = 0.8
##
##   Initial states:
##     l = 31216069.0517
##     b = 958394.2718
##
##   sigma:  8936314
##
##      AIC     AICc      BIC
## 2209.451 2209.872 2215.783
```

```
fit5$model
```

```
## Damped Holt's method with exponential trend
##
## Call:
##  holt(x = TotalYearlyExports_ts, h = 12, damped = TRUE, initial = "simple",
##
##  Call:
##      exponential = TRUE, alpha = 0.6, beta = 0.2)
##
##   Smoothing parameters:
##     alpha = 0.6
##     beta  = 0.2
##     phi   = 0.8
##
##   Initial states:
##     l = 31222778.2224
##     b = 1.0521
##
##   sigma:  0.2142
##
##      AIC     AICc      BIC
## 2189.185 2189.606 2195.518
```

2. Verify accuracy against historical data:

```
accuracy(fit1, TotalYearlyExports_vector)
```

```
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -485271.6  9014603  6320774 -5.272938 20.60878 0.9913178
## Test set     9951246.4 13358513 10816687 24.353096 27.64625 1.6964337
##                    ACF1
## Training set -0.06052178
## Test set              NA
```

```
accuracy(fit2, TotalYearlyExports_vector)
```

```
##                    ME    RMSE     MAE       MPE     MAPE      MASE
## Training set -1241747 8908094 6639236 -7.912605 21.45472 1.0412637
## Test set      2216450 7201940 6070544  2.799430 16.80889 0.9520731
##                    ACF1
## Training set 0.06702163
## Test set             NA
```

```
accuracy(fit3, TotalYearlyExports_vector)
```

```
##                     ME    RMSE     MAE       MPE     MAPE      MASE
## Training set  -347662.7 6985125 6047879 -4.707081 18.08399 0.9485183
## Test set     -1582508.6 6556149 5639031 -8.070011 17.35911 0.8843967
##                    ACF1
## Training set 0.1776038
## Test set             NA
```

```
accuracy(fit4, TotalYearlyExports_vector)
```

```
##                    ME     RMSE      MAE      MPE     MAPE     MASE
## Training set -145153  8936314  6570674 -4.20460 21.35029 1.030511
## Test set     9817917 12797426 10518538 24.32535 26.99133 1.649673
##                    ACF1
## Training set -0.0822064
## Test set             NA
```

```
accuracy(fit5, TotalYearlyExports_vector)
```

```
##                      ME    RMSE     MAE       MPE     MAPE     MASE
## Training set -575254.3 8951861 6982417 -5.973255 22.35146 1.095086
## Test set     4107072.6 8597312 7160245  7.946777 19.21703 1.122976
##                    ACF1
## Training set 0.04994958
## Test set             NA
```

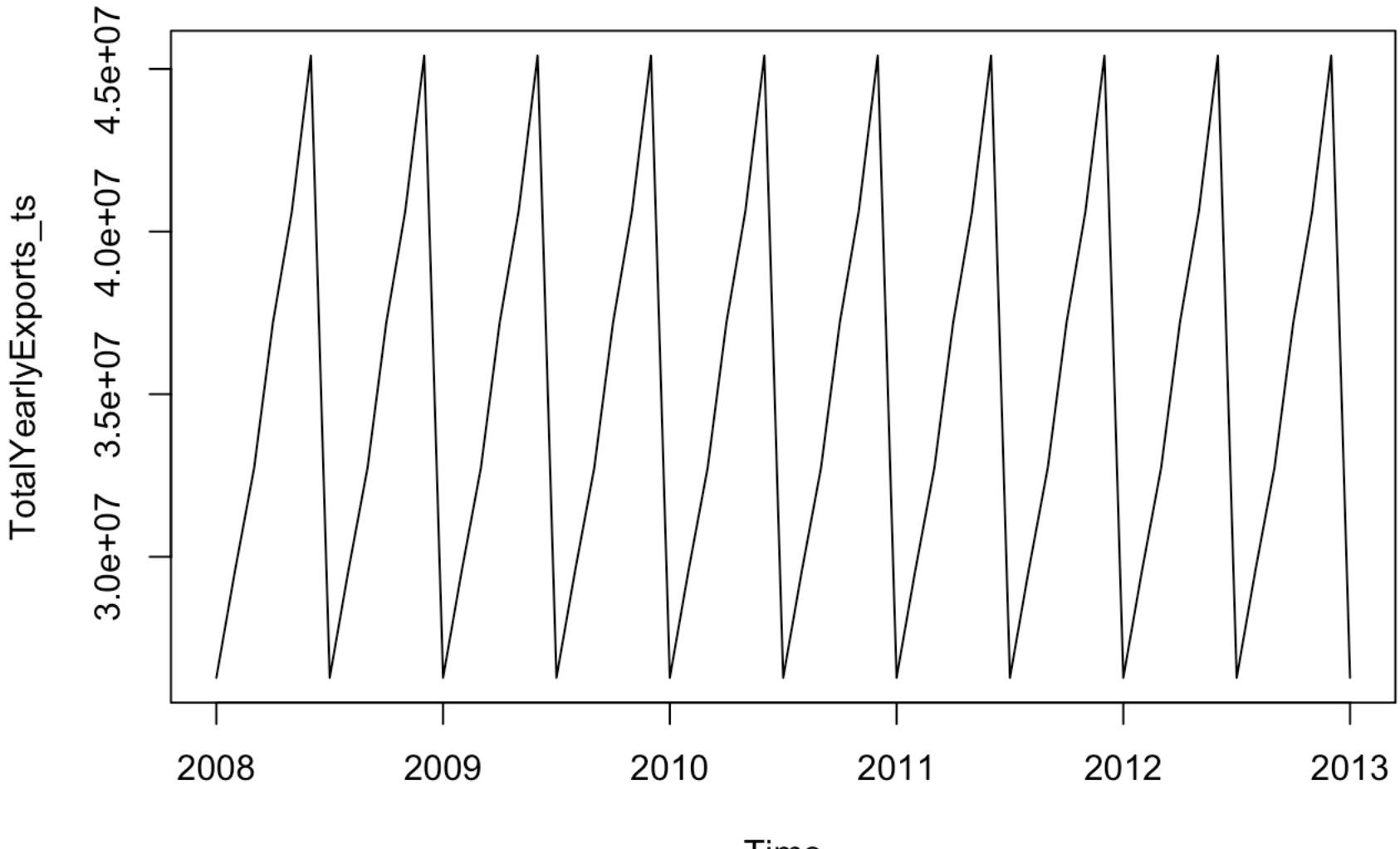
3. Conclusion: Our previous findings match those obtained with the accuracy results. Notice F3(defaults) has the lowest values for ERROR in the testing Set. Also Fit5(Damped Holt's method with exponential trend) seems to have pretty low values.

Holt Winter Seasonal Method

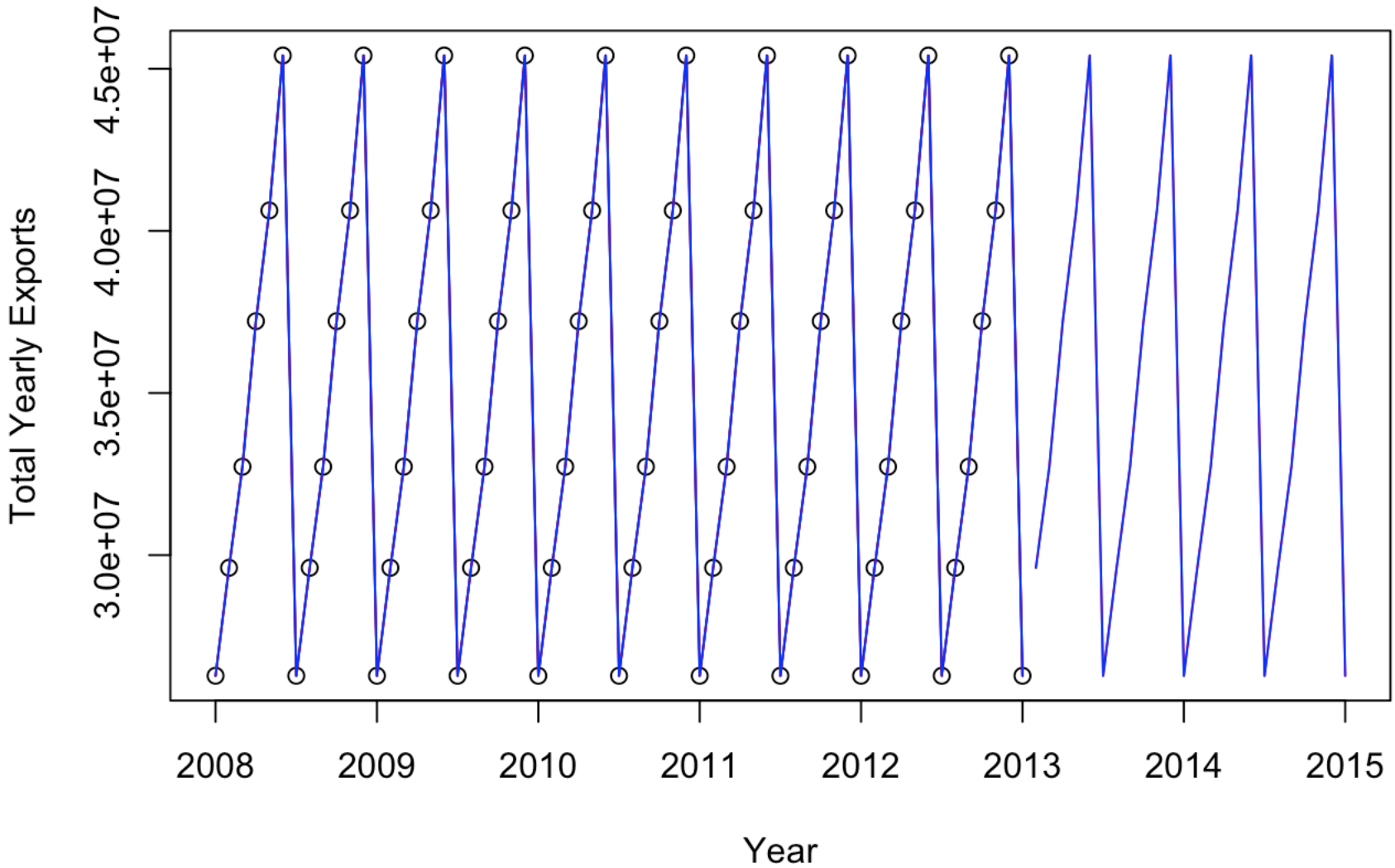
This method also offers a damped version and it can be additive (seasonal variations roughly constant through the series) or exponential (seasonal variations changing proportional to the level of series).

Lastly we would like to run the Holt Winter Seasonal Method but using the hw function as seen in the module 9 videos.

```
plot(TotalYearlyExports_ts, type="l")
```




```
fit1 <- hw(TotalYearlyExports_ts, seasonal="additive")
fit2 <- hw(TotalYearlyExports_ts, seasonal="multiplicative")
plot(fit2, ylab="Total Yearly Exports", plot.conf=FALSE, type="o", fcol="white", xlab
="Year", main = "Comparing Seasonal Methods")
lines(fitted(fit1), col="red")
lines(fitted(fit2), col="blue")
lines(fit1$mean, col="red")
lines(fit2$mean, col="blue")
```




```
fit1$mean
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 2013          29609916 32726772 37215503 40629676 45408410 26280011
## 2014 26280011 29609916 32726772 37215503 40629676 45408410 26280011
## 2015 26280011
##           Aug      Sep      Oct      Nov      Dec
## 2013 29609916 32726772 37215503 40629676 45408410
## 2014 29609916 32726772 37215503 40629676 45408410
## 2015
```

```
fit2$mean
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 2013          29609916 32726772 37215503 40629676 45408410 26280011
## 2014 26280011 29609916 32726772 37215503 40629676 45408410 26280011
## 2015 26280011
##           Aug      Sep      Oct      Nov      Dec
## 2013 29609916 32726772 37215503 40629676 45408410
## 2014 29609916 32726772 37215503 40629676 45408410
## 2015
```

```
accuracy(fit1)
```

```
##                         ME         RMSE          MAE           MPE
## Training set -2.748165e-09 8.782033e-09 3.236728e-09 -9.035265e-15
##                      MAPE MASE      ACF1
## Training set 1.011119e-14  Inf 0.2139382
```

```
accuracy(fit2)
```

```
##                         ME         RMSE          MAE           MPE
## Training set -4.641345e-09 1.385691e-08 6.595596e-09 -1.299667e-14
##                      MAPE MASE       ACF1
## Training set 1.875933e-14  Inf -0.1895224
```

Conclusion: At first it looks like either one of the Holt Winter Methods with seasonal component (additive or exponential) used in the cases above fits a very good forecast to what historical data looks like, however when looking closer at the results returned by the accuracy function it seems the multiplicative model is slighly better over time as the values for ERROR a lower for the Test set.

Comparing vs Planned Data

---

1. What is the best model for the export data? It is hard to decide. In fact we were able to obtain what seemed to b pretty good forecast numbers using various models. For example both Arimo(2,0,0) and Holt Winters Multiplicative were able to consider seasonal component as part of the parameters and thus making the results more accurate for our specific dataset.
2. How do you define “best”? Definition of “best” is rather relative to the components observed during time series data decomposition. It appears there is no one model better than others when it comes to forecasting data, instead components such as Seasonality, Trend, etc define the needs (data requirements to be considered) within the different parameters when conforming a specific model to used at the time of forecast. We like Arima Model in the sense that we were able to tweak the parameters to accomodate for more accurate results. Also Holt Winters Model seemed better fit for our particular dataset.

3. Looking at our actual predicted values using Holt Winters Filtering:

```
forecast_exports
```

```
##    forecast_exports
## 1          29609916
## 2          32726772
## 3          37215503
## 4          40629676
## 5          45408410
## 6          26280011
## 7          29609916
## 8          32726772
## 9          37215503
## 10         40629676
## 11         45408410
## 12         26280011
```

4. Looking at our actual predicted values using Holt Winters Multiplicative with Seasonal Component:

```
fitted(fit2)
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 2008 26280011 29609916 32726772 37215503 40629676 45408410 26280011
## 2009 26280011 29609916 32726772 37215503 40629676 45408410 26280011
## 2010 26280011 29609916 32726772 37215503 40629676 45408410 26280011
## 2011 26280011 29609916 32726772 37215503 40629676 45408410 26280011
## 2012 26280011 29609916 32726772 37215503 40629676 45408410 26280011
## 2013 26280011
##           Aug      Sep      Oct      Nov      Dec
## 2008 29609916 32726772 37215503 40629676 45408410
## 2009 29609916 32726772 37215503 40629676 45408410
## 2010 29609916 32726772 37215503 40629676 45408410
## 2011 29609916 32726772 37215503 40629676 45408410
## 2012 29609916 32726772 37215503 40629676 45408410
## 2013
```

5. Compare to Planned Exports

```
tidyImportedPlanDataChulwalar
```

```
##                  TotalPlan     2008     2009     2010     2011     2012
## 1                      Jan  2243103  2547980  2965885  3113110  3895396
## 2                      Feb  2162705  2247049  2751170  2883766  3588151
## 3                      Mar  2720911  2731156  2906493  2957893  3787240
## 4                      Apr  2011182  2020158  2383358  2601648  3036434
## 5                      May  1877757  2098038  2246893  2370949  2907891
```

```
##  6                    Jun  1819924  1927995  1992851  2339881  2707822
##  7                    Jul  1682196  1783692  2023434  2105328  2619486
##  8                    Aug  1893171  1907705  2244997  2341623  3784557
##  9                    Sep  3325711  3124040  3257717  4086297  4987460
## 10                    Oct  2662148  3102251  3536338  3640827  4367319
## 11                    Nov  2909966  3154669  3358206  3502334  4205772
## 12                    Dec  2574633  2742367  3112906  3280476  4059533
## 13                           NA       NA       NA       NA       NA
## 14          Coffee Plan      2008     2009     2010     2011     2012
## 15                    Jan   492421   450498   506991   646987  1057786
## 16                    Feb   444995   380959   550412   652598  1006335
## 17                    Mar   665274   592616   629309   778405  1260206
## 18                    Apr   444369   400839   468600   717677  1006509
## 19                    May   487668   471523   535435   684701   979754
## 20                    Jun   445242   405564   475326   639433   985549
## 21                    Jul   443318   401100   482147   659271   964181
## 22                    Aug   501222   444250   466887   652132  1027988
## 23                    Sep   546249   488899   532164   736826  1090561
## 24                    Oct   553286   584729   543650   774047  1151231
## 25                    Nov   664734   659061   662090   791780  1222188
## 26                    Dec   560104   512219   527275   823396  1148541
## 27                           NA       NA       NA       NA       NA
## 28          Spices Plan      2008     2009     2010     2011     2012
## 29                    Jan   424190   443454   685504   593024   665434
## 30                    Feb   388688   381571   559040   570173   657383
## 31                    Mar   457796   471631   590397   552269   706987
## 32                    Apr   363828   393075   566135   522050   601083
## 33                    May   364246   379443   448967   458092   604292
## 34                    Jun   358439   360120   442838   475669   571937
## 35                    Jul   321255   337682   423206   451094   575704
## 36                    Aug   370153   381164   458609   602954   802634
## 37                    Sep   645618   597557   651525   751102   911343
## 38                    Oct   470648   511889   598009   736236   830770
## 39                    Nov   529375   573453   575012   681492   814818
## 40                    Dec   448355   478396   544435   693967   870857
## 41                           NA       NA       NA       NA       NA
## 42    Tea Total Plan         2008     2009     2010     2011     2012
## 43                    Jan  1263613  1546801  1648769  1781991  2070256
## 44                    Feb  1231125  1378217  1490577  1564272  1731099
## 45                    Mar  1489621  1563799  1538493  1455531  1663266
## 46                    Apr  1051346  1166229  1208636  1257528  1232994
## 47                    May   933392  1057223  1104777  1134418  1164076
## 48                    Jun   932047   983279   931127  1018200  1018137
## 49                    Jul   855520   913751   916160   843336   932241
## 50                    Aug   923070   980703  1096933   974375  1800576
## 51                    Sep  2080877  1974166  1832882  2435674  2823873
## 52                    Oct  1575579  1886971  2103588  1972649  2224655
## 53                    Nov  1561956  1839155  1877929  1873075  2025003
## 54                    Dec  1515127  1727567  1862684  1684766  1955509
## 55                           NA       NA       NA       NA       NA
```

```
## 56          Loose Tea Plan     2008     2009     2010     2011     2012
## 57                    Jan   449227   394188   388677   412463   481147
## 58                    Feb   373663   320490   317587   323577   412798
## 59                    Mar   415732   351375   306376   313230   364106
## 60                    Apr   331337   271021   275940   276210   311291
## 61                    May   290942   225914   235850   249768   283279
## 62                    Jun   287603   234600   224371   217911   286839
## 63                    Jul   245390   191342   204869   209229   249233
## 64                    Aug   284540   226507   220570   219002   288342
## 65                    Sep   554127   519935   357203   365415   399167
## 66                    Oct   467772   512283   413862   421679   524838
## 67                    Nov   469089   456203   357645   359800   399038
## 68                    Dec   409962   376595   364243   343171   415564
## 69                           NA       NA       NA       NA       NA
## 70          Teabag Plan     2008     2009     2010     2011     2012
## 71                    Jan   814386  1152613  1260092  1369528  1589109
## 72                    Feb   857462  1057727  1172990  1240695  1318301
## 73                    Mar  1073889  1212424  1232117  1142301  1299159
## 74                    Apr   720009   895208   932696   981318   921703
## 75                    May   642450   831309   868927   884650   880796
## 76                    Jun   644444   748679   706756   800289   731299
## 77                    Jul   610130   722409   711291   634107   683008
## 78                    Aug   638530   754196   876363   755372  1512234
## 79                    Sep  1526750  1454231  1475679  2070259  2424705
## 80                    Oct  1107807  1374688  1689726  1550970  1699817
## 81                    Nov  1092867  1382952  1520284  1513274  1625965
## 82                    Dec  1105165  1350972  1498441  1341595  1539945
## 83                           NA       NA       NA       NA       NA
## 84 Total yearly sales Plan     2008     2009     2010     2011     2012
## 85                    Jan 27883407 29387100 32780247 35224132 43947063
## 86                    Feb 27883407 29387100 32780247 35224132 43947063
## 87                    Mar 27883407 29387100 32780247 35224132 43947063
## 88                    Apr 27883407 29387100 32780247 35224132 43947063
## 89                    May 27883407 29387100 32780247 35224132 43947063
## 90                    Jun 27883407 29387100 32780247 35224132 43947063
## 91                    Jul 27883407 29387100 32780247 35224132 43947063
## 92                    Aug 27883407 29387100 32780247 35224132 43947063
## 93                    Sep 27883407 29387100 32780247 35224132 43947063
## 94                    Oct 27883407 29387100 32780247 35224132 43947063
## 95                    Nov 27883407 29387100 32780247 35224132 43947063
## 96                    Dec 27883407 29387100 32780247 35224132 43947063
##        2013    2014
## 1   3580325 4474000
## 2   3863212 4185565
## 3   3606083 4278119
## 4   3213575 3985542
## 5   3139128 3605973
## 6   2998610 3515173
## 7   2785453 3269444
## 8   3083654 3656112
```

```
## 9   5143757 5637391
## 10  4149334 5157781
## 11  4495212 5353458
## 12  4093664 4703185
## 13       NA      NA
## 14     2013      NA
## 15   940156      NA
## 16  1094548      NA
## 17  1053751      NA
## 18  1072364      NA
## 19  1061436      NA
## 20  1077276      NA
## 21   984463      NA
## 22  1010619      NA
## 23  1083541      NA
## 24  1089769      NA
## 25  1151019      NA
## 26  1044125      NA
## 27       NA      NA
## 28     2013      NA
## 29   670157      NA
## 30   673123      NA
## 31   727908      NA
## 32   680251      NA
## 33   687880      NA
## 34   702883      NA
## 35   623366      NA
## 36   694089      NA
## 37  1029222      NA
## 38   853935      NA
## 39   889003      NA
## 40   842765      NA
## 41       NA      NA
## 42     2013      NA
## 43  1864733      NA
## 44  1837228      NA
## 45  1663834      NA
## 46  1305603      NA
## 47  1172373      NA
## 48  1089115      NA
## 49  1074687      NA
## 50  1217930      NA
## 51  2916115      NA
## 52  2043888      NA
## 53  2199880      NA
## 54  2133214      NA
## 55       NA      NA
## 56     2013      NA
## 57   360982      NA
## 58   342370      NA
```

```
## 59   346868      NA
## 60   277548      NA
## 61   251623      NA
## 62   257153      NA
## 63   232752      NA
## 64   252611      NA
## 65   494843      NA
## 66   445720      NA
## 67   414612      NA
## 68   401854      NA
## 69       NA      NA
## 70     2013      NA
## 71  1503751      NA
## 72  1494858      NA
## 73  1316966      NA
## 74  1028055      NA
## 75   920750      NA
## 76   831961      NA
## 77   841936      NA
## 78   965319      NA
## 79  2421272      NA
## 80  1598167      NA
## 81  1785268      NA
## 82  1731360      NA
## 83       NA      NA
## 84     2013      NA
## 85 44152007      NA
## 86 44152007      NA
## 87 44152007      NA
## 88 44152007      NA
## 89 44152007      NA
## 90 44152007      NA
## 91 44152007      NA
## 92 44152007      NA
## 93 44152007      NA
## 94 44152007      NA
## 95 44152007      NA
## 96 44152007      NA
```

```
TotalYearlyExports_Planned <- tidyImportedPlanDataChulwalar[84:97,]
TotalYearlyExports_Planned <- TotalYearlyExports_Planned[c(1:7)]
TotalYearlyExports_Planned
```

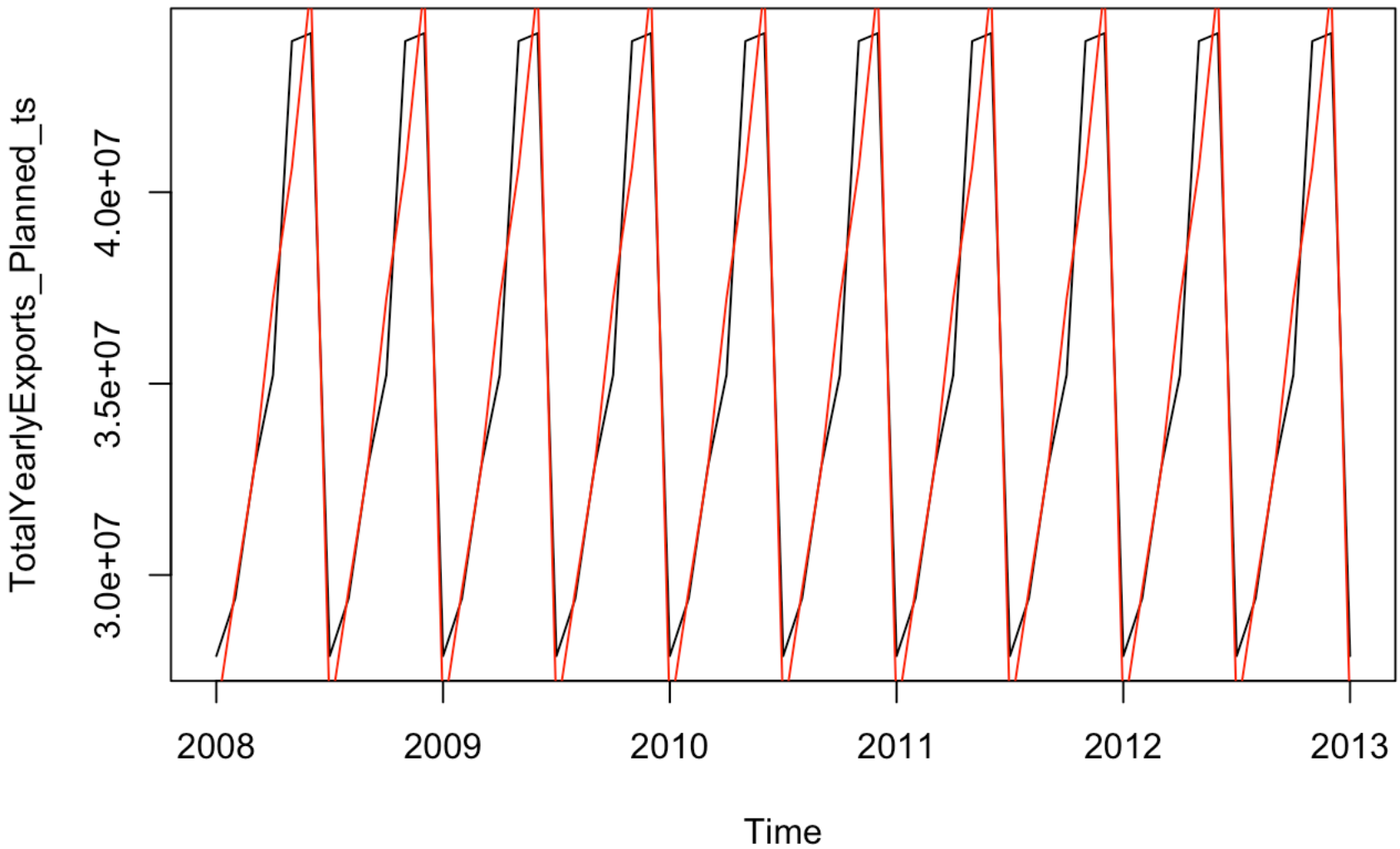
```
##                   TotalPlan     2008     2009     2010     2011     2012
## 84 Total yearly sales Plan     2008     2009     2010     2011     2012
## 85                      Jan 27883407 29387100 32780247 35224132 43947063
## 86                      Feb 27883407 29387100 32780247 35224132 43947063
## 87                      Mar 27883407 29387100 32780247 35224132 43947063
## 88                      Apr 27883407 29387100 32780247 35224132 43947063
## 89                      May 27883407 29387100 32780247 35224132 43947063
## 90                      Jun 27883407 29387100 32780247 35224132 43947063
## 91                      Jul 27883407 29387100 32780247 35224132 43947063
## 92                      Aug 27883407 29387100 32780247 35224132 43947063
## 93                      Sep 27883407 29387100 32780247 35224132 43947063
## 94                      Oct 27883407 29387100 32780247 35224132 43947063
## 95                      Nov 27883407 29387100 32780247 35224132 43947063
## 96                      Dec 27883407 29387100 32780247 35224132 43947063
## NA                     <NA>       NA       NA       NA       NA       NA
##        2013
## 84     2013
## 85 44152007
## 86 44152007
## 87 44152007
## 88 44152007
## 89 44152007
## 90 44152007
## 91 44152007
## 92 44152007
## 93 44152007
## 94 44152007
## 95 44152007
## 96 44152007
## NA       NA
```

```
TotalYearlyExports_Planned_vector <- c(27883407,29387100,32780247,35224132,43947063,4
4152007)
TotalYearlyExports_Planned_vector
```

```
## [1] 27883407 29387100 32780247 35224132 43947063 44152007
```

6. Plot Actual Planned vs Forecast

```
TotalYearlyExports_Planned_ts <- ts(TotalYearlyExports_Planned_vector, start=c(2008),
end=c(2013), frequency=12)
plot.ts(TotalYearlyExports_Planned_ts)
lines(fitted(fit2), col="red")
```



****

7. Coclusion: Holt Winters Multiplicative with Seasonal Component is our best model. Its Forecast values, although a bit more positive, are much closer to those planned.

```
fit2$model
```

```
## Holt-Winters' multiplicative method
##
## Call:
##  hw(x = TotalYearlyExports_ts, seasonal = "multiplicative")
##
##   Smoothing parameters:
##     alpha = 0.0418
##     beta  = 0.0043
##     gamma = 0.8636
##
##   Initial states:
##     l = 35311714.6667
##     b = 0
##     s=1.2859 1.1506 1.0539 0.9268 0.8385 0.7442
##            1.2859 1.1506 1.0539 0.9268 0.8385 0.7442
##
##   sigma:  0
##
##       AIC      AICc       BIC
## -1926.392 -1914.028 -1892.618
```

Additional Anaylisis

Further analysis on this case can be found on https://github.com/kkillion43/Unit10CaseStudy2 (https://github.com/kkillion43/Unit10CaseStudy2)

R Data Science Essentials, Ravindran Sharan Kumar, Raja B. Koushik, Packt Publishing, January 2016