Asylum Seekers Entering the EU

1. Exploratory Data Analysis

1.1. Data Source, Headers and Labels

DATASET "Asylum and first time asylum applicants by citizenship, age and sex Monthly data (rounded) [migr_asyappctzm]"
LAST UPDATE* 19.03.19 06:51:45
EXTRACTION DATE 19.03.19 13:19:35
SOURCE OF DATA Eurostat

Attribute	Values
TIME	"2008M01" to "2018M12"
GEO	"European Union - 28 countries"
CITIZEN	"Extra-EU28"
SEX	"Total", "Males", "Females"
AGE	"Total"
ASYL_APP	"Asylum applicant", "First time applicant"
UNIT"	"Person"

No footnotes available.

Available flags:

- b,"break in time series"
- c,"confidential"
- d,"definition differs, see metadata"
- e,"estimated"
- f,"forecast"
- n,"not significant"
- p,"provisional"
- r,"revised"
- s,"Eurostat estimate"
- u,"low reliability"
- z,"not applicable"
- Special value: ":","not available"

1.2 Reading the data

```
#set working directory
setwd("~/DIT/Time Series 2 Forecasting/Project/2-Trend and seasonality")

#Graphical Parameters: For colours, color specifications, check colors() or, even better, demo(colors)
```

```r
library(readr)
library(forecast)
library(tseries)
library(mgcv)
```

```
## Loading required package: nlme

##
## Attaching package: 'nlme'

## The following object is masked from 'package:forecast':
##
##     getResponse

## This is mgcv 1.8-26. For overview type 'help("mgcv-package")'.
```

```r
datafile <- read_csv("migr_asylum_applicant_EU28/migr_asyappctzm_1_Data.csv")
```

```
## Parsed with column specification:
## cols(
##   TIME = col_character(),
##   DATE_PATTERN = col_character(),
##   GEO = col_character(),
##   CITIZEN = col_character(),
##   SEX = col_character(),
##   AGE = col_character(),
##   ASYL_APP = col_character(),
##   UNIT = col_character(),
##   Value = col_number(),
##   `Flag and Footnotes` = col_character()
## )
```

```r
head(datafile)
```

```
## # A tibble: 6 x 10
##   TIME  DATE_PATTERN GEO   CITIZEN SEX   AGE   ASYL_APP UNIT  Value
##   <chr> <chr>        <chr> <chr>   <chr> <chr> <chr>    <chr> <dbl>
## 1 2008~ 01/01/08     Euro~ Extra-~ Total Total Asylum ~ Pers~ 18355
## 2 2008~ 01/02/08     Euro~ Extra-~ Total Total Asylum ~ Pers~ 16885
## 3 2008~ 01/03/08     Euro~ Extra-~ Total Total Asylum ~ Pers~ 15570
## 4 2008~ 01/04/08     Euro~ Extra-~ Total Total Asylum ~ Pers~ 16975
## 5 2008~ 01/05/08     Euro~ Extra-~ Total Total Asylum ~ Pers~ 17110
## 6 2008~ 01/06/08     Euro~ Extra-~ Total Total Asylum ~ Pers~ 17520
## # ... with 1 more variable: `Flag and Footnotes` <chr>
```

All column names:

```r
names(datafile)
```

```
##  [1] "TIME"               "DATE_PATTERN"       "GEO"
##  [4] "CITIZEN"            "SEX"                "AGE"
##  [7] "ASYL_APP"           "UNIT"               "Value"
## [10] "Flag and Footnotes"
```

Convert the values into a time series Plot Values against Time

```r
values = datafile[9]
values = ts(values, start=c(2008, 1), end=c(2018, 12), frequency=12)
ts.plot(values, main="Asylum Applications to EU28 Countries 2008-2018", ylab="Number of Applications")
```


```r
summary(values)
```

```
##      Value
##  Min.   : 15570
##  1st Qu.: 22881
##  Median : 35925
##  Mean   : 48345
##  3rd Qu.: 60488
##  Max.   :171895
```

```r
end(values)
```

```
## [1] 2018   12
```

---

2. Smoothing the data

We can use smoothing to reduce the volatility in our observed data and making it into a more stable and predictable series.

Trend Analysis - non-parametric

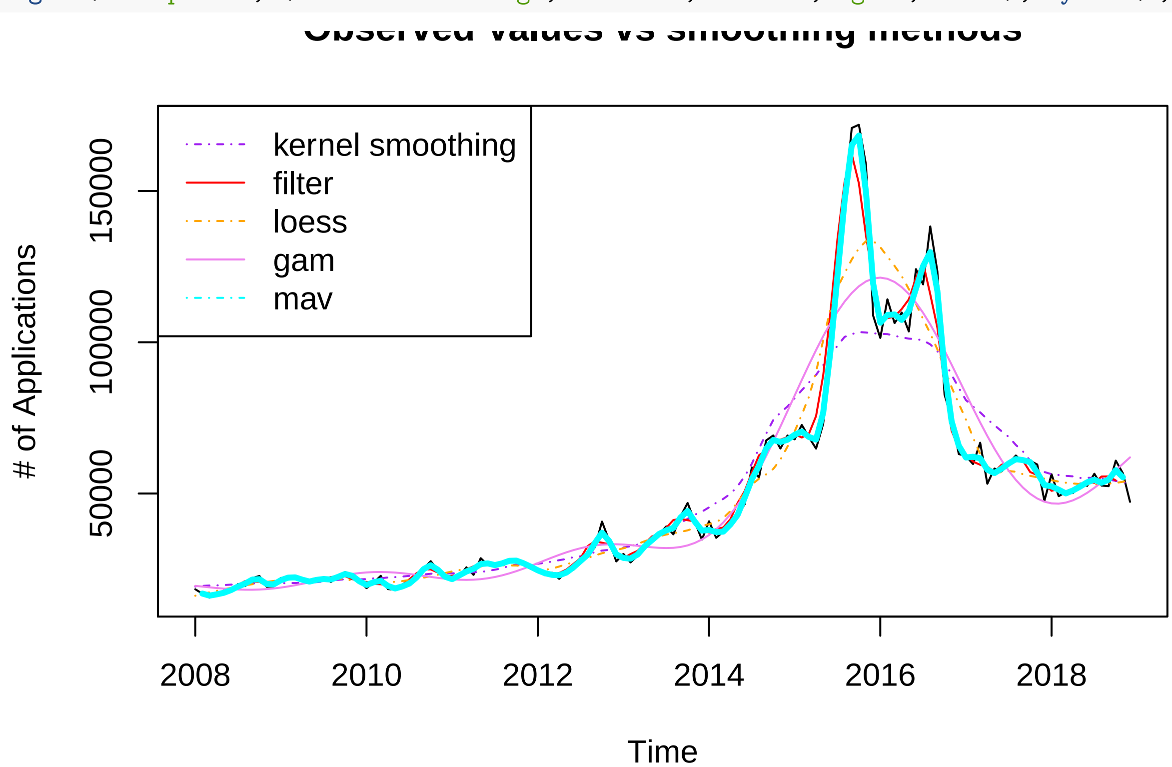
```r
#Create equally spaced time points for fitting trends
time.pts = c(1:length(values))
time.pts = c(time.pts - min(time.pts))/max(time.pts)
```

Fitting a moving average, kernel smoothing, loess and splines smoothing

```
#define mav/smoothing methods, and fit
values.mafilter.fit = filter(values, filter = rep(1/4, 4), sides = 2)


ma.fit = ma(values, order=2, centre=TRUE)
values.fit.ma = ts(ma.fit, start=c(2008, 1), frequency=12)

ksmooth.fit = ksmooth(time.pts, values, kernel = "box", bandwidth = 0.2)
values.fit.ksmooth = ts(ksmooth.fit$y,start=c(2008, 1),frequency=12)

loess.fit = loess(as.matrix(values)~time.pts, data=values, span=0.2)
values.fit.loess = ts(predict(loess.fit), start=c(2008, 1), frequency=12)

gam.fit = gam(values~s(time.pts))
values.fit.gam = ts(fitted(gam.fit),start=c(2008, 1),frequency=12)
```

Plotting

```
# plot fits against values
ts.plot(values,ylab="# of Applications", main="Observed Values vs smoothing methods")
lines(values.fit.ksmooth,lwd=1, lty=4 ,col="purple")
lines(values.mafilter.fit, col="red")
lines(values.fit.loess, col="orange", lty=4)
lines(values.fit.gam, col="violet")
lines(values.fit.ma, col="cyan", lwd=3)
legend(x="topleft",c("kernel smoothing", "filter", "loess", "gam", "mav"),lty = c(4, 1), col=c("purple"
```




```
#values.fit.mav is the ma dataframe (type of the objects is float) with the transformed values for each
#ablines is an a, b line graphing function, a is y intercept and b is slope
```

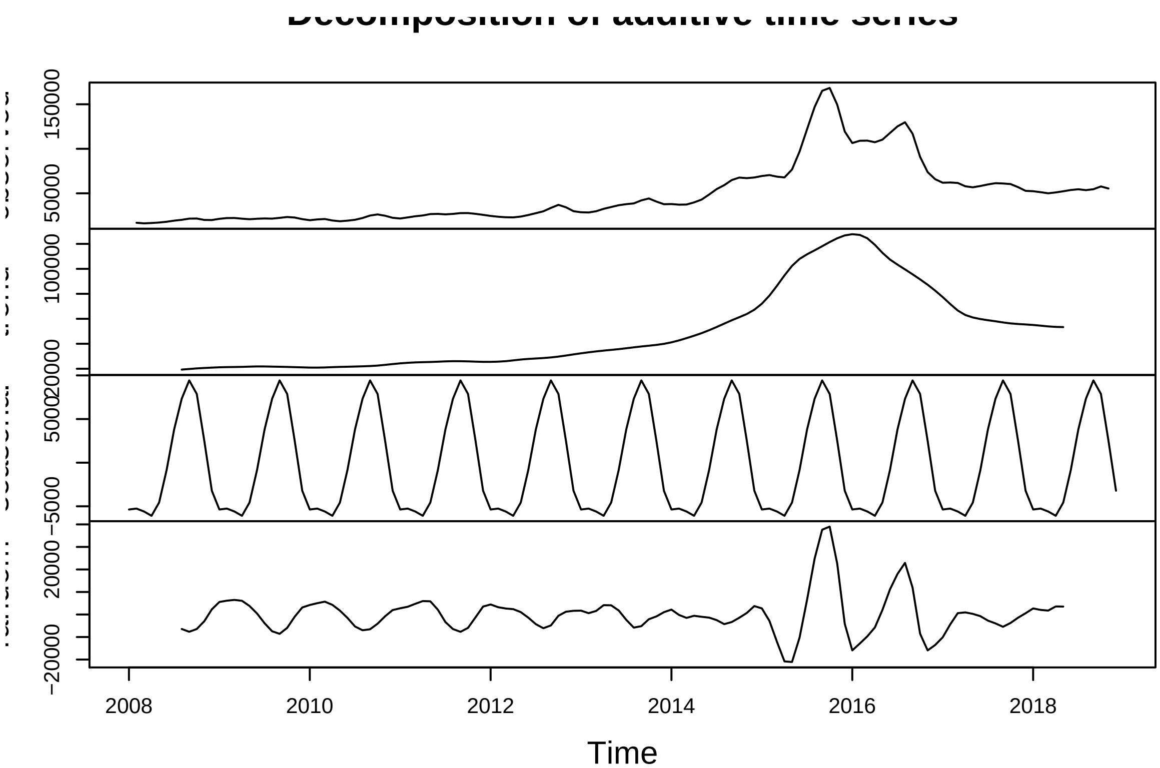
It's a matter of balancing smoothness to make predictability easier and accuracy. Pick filter(red) or moving average(cyan).

---

3. Decomposing the data

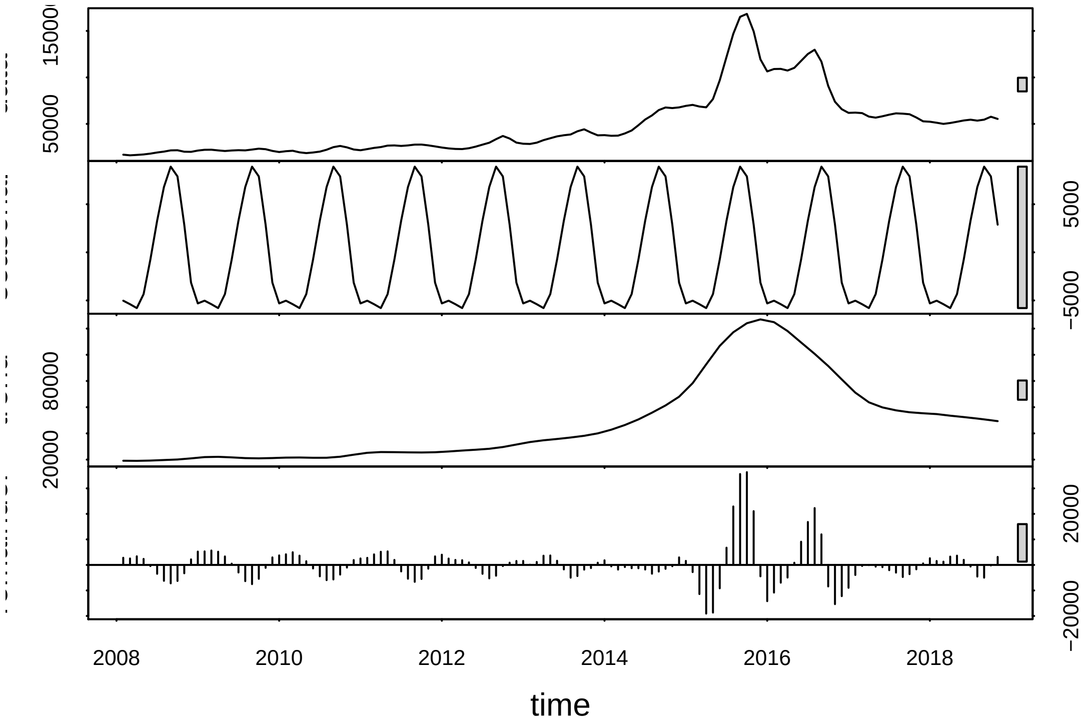
Trend component, seasonal component and residuals (random/white noise). We assume this series is additive (instead of multiplicative).

```r
values.ma.decomp = decompose(values.fit.ma, type=c("additive"))
plot(values.ma.decomp)
```



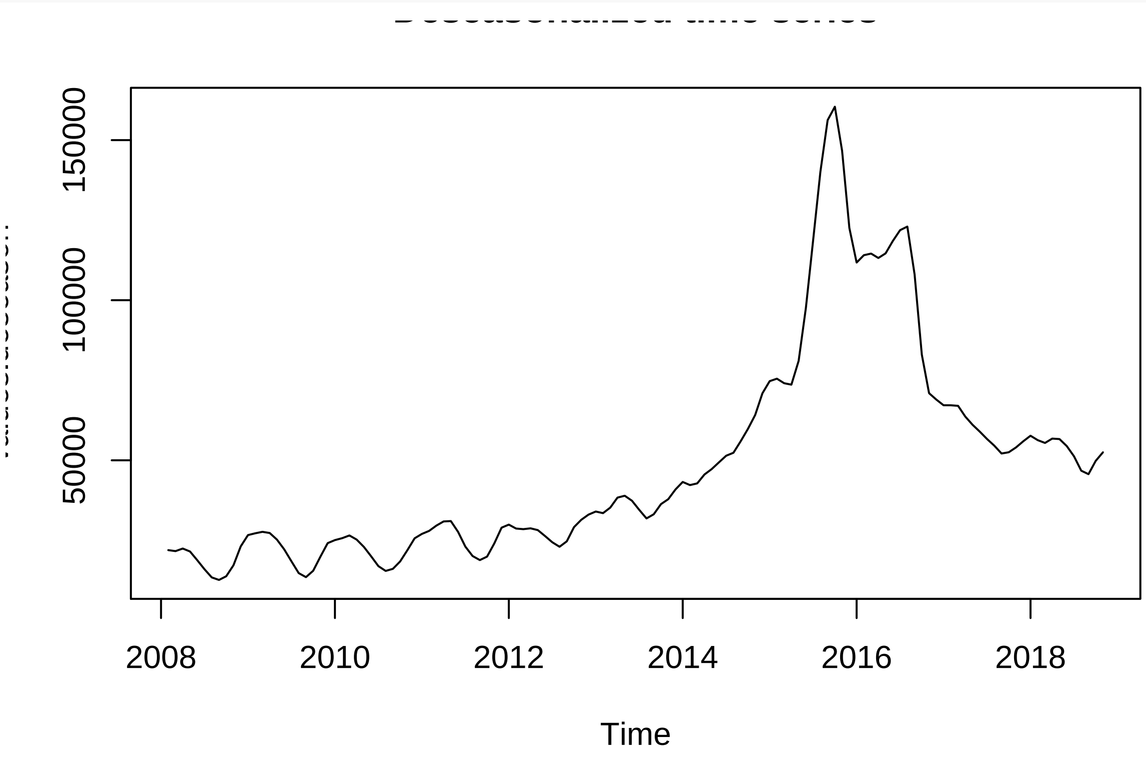
A more detailed decomposition of this time-series can be seen below by using STL - Seasonal Decomposition of Time Series by LOESS:

```r
values.decomp.stl <- stl(na.omit(values.fit.ma[, 1]), s.window="periodic")
plot(values.decomp.stl)
```



Using the decomposed time series, we can subtract the $seasonal component from the data and de-seasonalize it - which can be useful to apply an ARIMA process, instead of SARIMA.

```
values.deseason = seasadj(values.decomp.stl)
plot(values.deseason, main="Deseasonalized time series")
```



```r
print(values.decomp.stl)
```

```
##  Call:
##  stl(x = na.omit(values.fit.ma[, 1]), s.window = "periodic")
##
## Components
##             seasonal    trend   remainder
## Feb 2008 -5025.1317  19144.14   2804.74388
## Mar 2008 -5391.2925  19102.97   2538.32444
## Apr 2008 -5792.7941  19061.80   3388.49574
## May 2008 -4342.8346  19158.09   2363.49072
## Jun 2008  -747.0787  19254.39   -446.06072
## Jul 2008  3294.3252  19455.33  -3479.65891
## Aug 2008  6789.0248  19656.28  -6225.30284
## Sep 2008  8908.2448  19862.23  -7219.22019
## Oct 2008  7890.9851  20068.17  -6281.65786
## Nov 2008  2875.5202  20501.07  -3276.59209
## Dec 2008 -3149.1758  20933.97   2185.20483
## Jan 2009 -5309.7893  21417.41   5231.12483
## Feb 2009 -5025.1317  21900.86   5305.52362
## Mar 2009 -5391.2925  22007.61   5657.42787
## Apr 2009 -5792.7941  22114.37   5185.92286
## May 2009 -4342.8346  21900.41   3336.17913
## Jun 2009  -747.0787  21686.44    505.63897
## Jul 2009  3294.3252  21406.70  -2959.77100
## Aug 2009  6789.0248  21126.95  -6355.97673
## Sep 2009  8908.2448  21036.99  -7522.73152
## Oct 2009  7890.9851  20947.02  -5446.75664
## Nov 2009  2875.5202  21062.66  -1143.17845
## Dec 2009 -3149.1758  21178.29   2955.88090
## Jan 2010 -5309.7893  21350.19   3734.59544
## Feb 2010 -5025.1317  21522.09   4163.03877
## Mar 2010 -5391.2925  21560.08   4968.71193
## Apr 2010 -5792.7941  21598.07   3643.47582
## May 2010 -4342.8346  21497.63   1451.45609
## Jun 2010  -747.0787  21397.19  -1383.86007
## Jul 2010  3294.3252  21401.78  -4456.10535
## Aug 2010  6789.0248  21406.37  -5952.89639
## Sep 2010  8908.2448  21798.20  -5690.19802
## Oct 2010  7890.9851  22190.03  -3784.76998
## Nov 2010  2875.5202  22960.24  -1039.50804
## Dec 2010 -3149.1758  23730.44   1913.73505
## Jan 2011 -5309.7893  24458.01   2546.78199
## Feb 2011 -5025.1317  25185.57   2774.55772
## Mar 2011 -5391.2925  25467.31   4137.72807
## Apr 2011 -5792.7941  25749.05   5157.48916
## May 2011 -4342.8346  25714.60   5310.73723
## Jun 2011  -747.0787  25680.14   1959.43888
## Jul 2011  3294.3252  25597.47  -2550.54120
## Aug 2011  6789.0248  25514.79  -5410.06704
## Sep 2011  8908.2448  25471.23  -6628.22914
## Oct 2011  7890.9851  25427.68  -5506.16158
## Nov 2011  2875.5202  25518.63  -1464.15347
## Dec 2011 -3149.1758  25609.59   3354.58580
```

```
## Jan 2012 -5309.7893  25896.70   4015.58495
## Feb 2012 -5025.1317  26183.82   2507.56289
## Mar 2012 -5391.2925  26539.92   1961.37151
## Apr 2012 -5792.7941  26896.02   1859.27087
## May 2012 -4342.8346  27214.11    993.72576
## Jun 2012  -747.0787  27532.19  -1185.11579
## Jul 2012  3294.3252  27900.64  -3489.96780
## Aug 2012  6789.0248  28269.09  -5273.11556
## Sep 2012  8908.2448  28924.57  -4225.31221
## Oct 2012  7890.9851  29580.04   -438.52919
## Nov 2012  2875.5202  30539.16    885.31627
## Dec 2012 -3149.1758  31498.28   1544.64289
## Jan 2013 -5309.7893  32448.96   1562.07654
## Feb 2013 -5025.1317  33399.64    100.48899
## Mar 2013 -5391.2925  34067.22   1169.07426
## Apr 2013 -5792.7941  34734.79   3625.50026
## May 2013 -4342.8346  35232.27   3704.31585
## Jun 2013  -747.0787  35729.74   1639.83502
## Jul 2013  3294.3252  36289.18  -1763.50248
## Aug 2013  6789.0248  36848.61  -5000.13572
## Sep 2013  8908.2448  37505.37  -4353.61903
## Oct 2013  7890.9851  38162.14  -1846.87266
## Nov 2013  2875.5202  39090.97  -1238.99406
## Dec 2013 -3149.1758  40019.81    869.36571
## Jan 2014 -5309.7893  41425.16   1819.62759
## Feb 2014 -5025.1317  42830.51   -565.38172
## Mar 2014 -5391.2925  44613.99  -1821.44659
## Apr 2014 -5792.7941  46397.46   -832.17072
## May 2014 -4342.8346  48535.17  -1278.58590
## Jun 2014  -747.0787  50672.88  -1308.29750
## Jul 2014  3294.3252  53236.11  -1799.18324
## Aug 2014  6789.0248  55799.34  -3443.36473
## Sep 2014  8908.2448  58564.59  -2606.58688
## Oct 2014  7890.9851  61329.84  -1527.07936
## Nov 2014  2875.5202  64654.45   -506.22072
## Dec 2014 -3149.1758  67979.06   2950.11908
## Jan 2015 -5309.7893  73140.35   1583.19031
## Feb 2015 -5025.1317  78301.64  -2809.00967
## Mar 2015 -5391.2925  85495.95 -11415.90859
## Apr 2015 -5792.7941  92690.26 -19062.46677
## May 2015 -4342.8346  99738.51 -18679.42489
## Jun 2015  -747.0787 106786.76  -9163.42943
## Jul 2015  3294.3252 111991.18   6761.99792
## Aug 2015  6789.0248 117195.60  22912.87953
## Sep 2015  8908.2448 120633.84  35617.91227
## Oct 2015  7890.9851 124072.09  36356.92467
## Nov 2015  2875.5202 125546.09  21077.13936
## Dec 2015 -3149.1758 127020.09  -4464.66480
## Jan 2016 -5309.7893 125961.75 -14198.21533
## Feb 2016 -5025.1317 124903.42 -10839.53707
## Mar 2016 -5391.2925 121520.44  -6955.39849
## Apr 2016 -5792.7941 118137.46  -4953.41918
## May 2016 -4342.8346 113766.11    859.22025
## Jun 2016  -747.0787 109394.77   9107.31325
```

```
## Jul 2016  3294.3252 105054.60  16826.07286
## Aug 2016  6789.0248 100714.44  22281.53671
## Sep 2016  8908.2448  96079.49  11978.52007
## Oct 2016  7890.9851  91444.53  -8428.01689
## Nov 2016  2875.5202  86320.69 -15361.20755
## Dec 2016 -3149.1758  81196.84 -12228.91705
## Jan 2017 -5309.7893  76160.67  -8963.38214
## Feb 2017 -5025.1317  71124.50  -3939.36845
## Mar 2017 -5391.2925  67424.06   -402.77190
## Apr 2017 -5792.7941  63723.63    -63.33462
## May 2017 -4342.8346  61776.12   -677.03061
## Jun 2017  -747.0787  59828.60   -877.77302
## Jul 2017  3294.3252  58709.95  -2058.02039
## Aug 2017  6789.0248  57591.29  -3014.06351
## Sep 2017  8908.2448  56858.01  -4731.25440
## Oct 2017  7890.9851  56124.73  -3610.71563
## Nov 2017  2875.5202  55742.38  -1715.40447
## Dec 2017 -3149.1758  55360.04    580.38785
## Jan 2018 -5309.7893  55053.38   2623.90565
## Feb 2018 -5025.1317  54746.73   1542.15226
## Mar 2018 -5391.2925  54132.26   1294.03268
## Apr 2018 -5792.7941  53517.79   3268.75384
## May 2018 -4342.8346  52991.30   3645.28718
## Jun 2018  -747.0787  52464.80   1993.52409
## Jul 2018  3294.3252  51894.89   -632.96836
## Aug 2018  6789.0248  51324.98  -4564.00655
## Sep 2018  8908.2448  50680.40  -5004.89305
## Oct 2018  7890.9851  50035.81   -213.04987
## Nov 2018  2875.5202  49350.87   3146.10821
```

The additive seasonal coefficients for each month are:

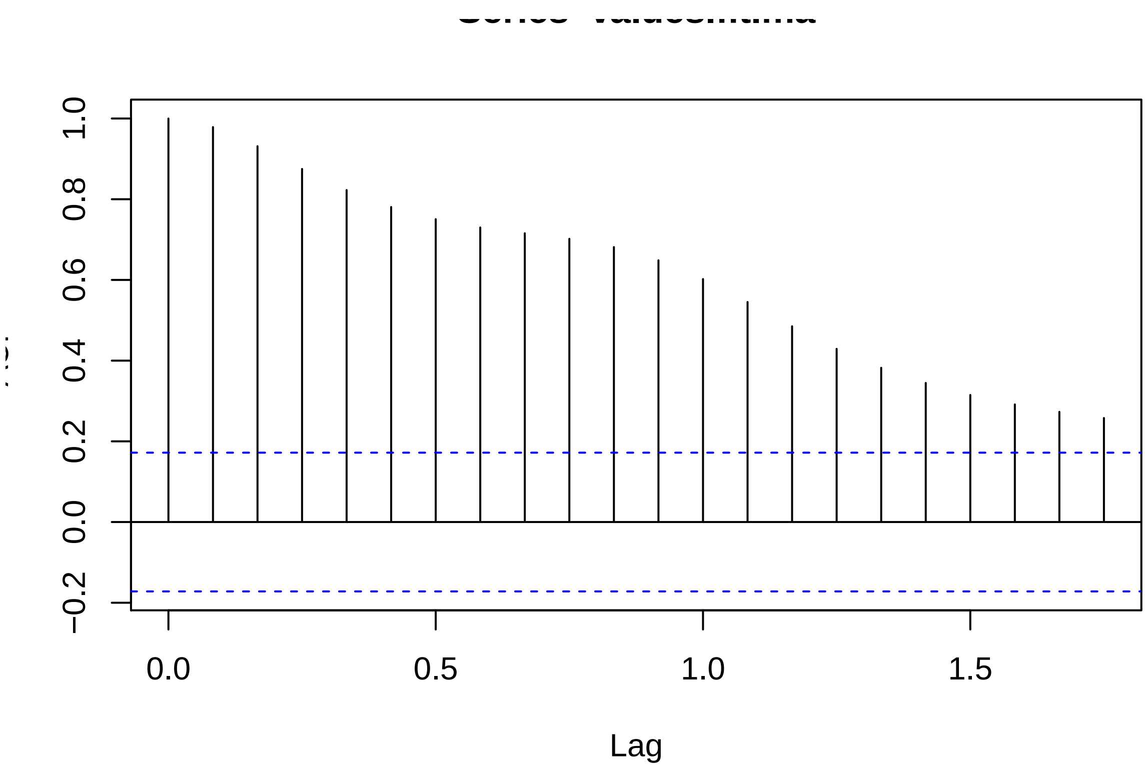
Jan -5309.7893
Feb -5025.1317
Mar -5391.2925
Apr -5792.7941
May -4342.8346
Jun -747.0787
Jul 3294.3252
Aug 6789.0248
Sep 8908.2448
Oct 7890.9851
Nov 2875.5202
Dec -3149.1758

---

4. Is the data stationary?

> "If we fit a stationary model to data, we assume our data are a realization of a stationary process. So our first step in an analysis should be to check whether there is any evidence of a trend or seasonal effects and, if there is, remove them."

```
values.fit.ma <- ts(na.omit(values.fit.ma), frequency=12, start=c(2008, 1))
acf(values.fit.ma)
```




```r
#adf.test(values.fit.ma)
```

The data doesn´t appear to be stationary - it is apparent that mean and variance change over time. Looking at the ACF plot we can also reach the same conclusion, given there is no sharp drop in the values, but instead a smooth decay. So we need to transform our data to make it more stationary in order to use it for ARIMA models, for instance. This is further confirmed by running the ADF test: with a p-value = 0.7279 we can't reject the null hypothesis of non-stationarity.

```r
adf.test(values.fit.ma)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  values.fit.ma
## Dickey-Fuller = -1.6359, Lag order = 5, p-value = 0.7279
## alternative hypothesis: stationary
```

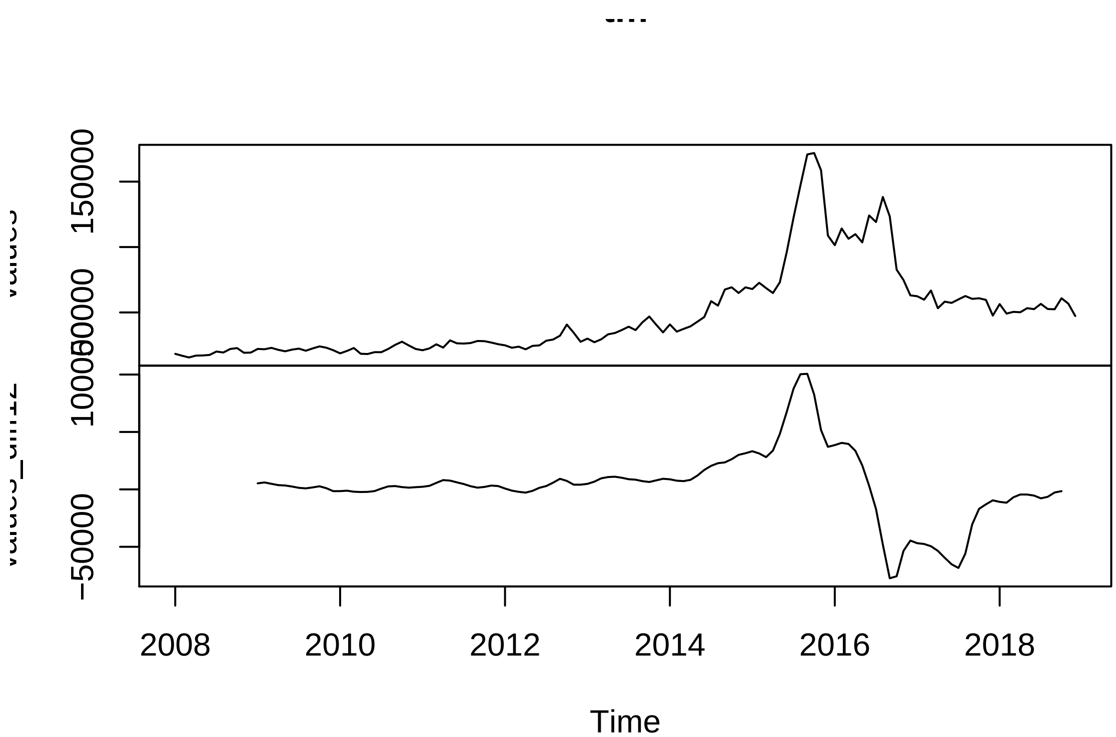
Making the data stationary

Some of the possible mathematical transforms include: differencing, log (and Box-Cox), moving average, percent change, lag, or cumulative sum.

Differencing

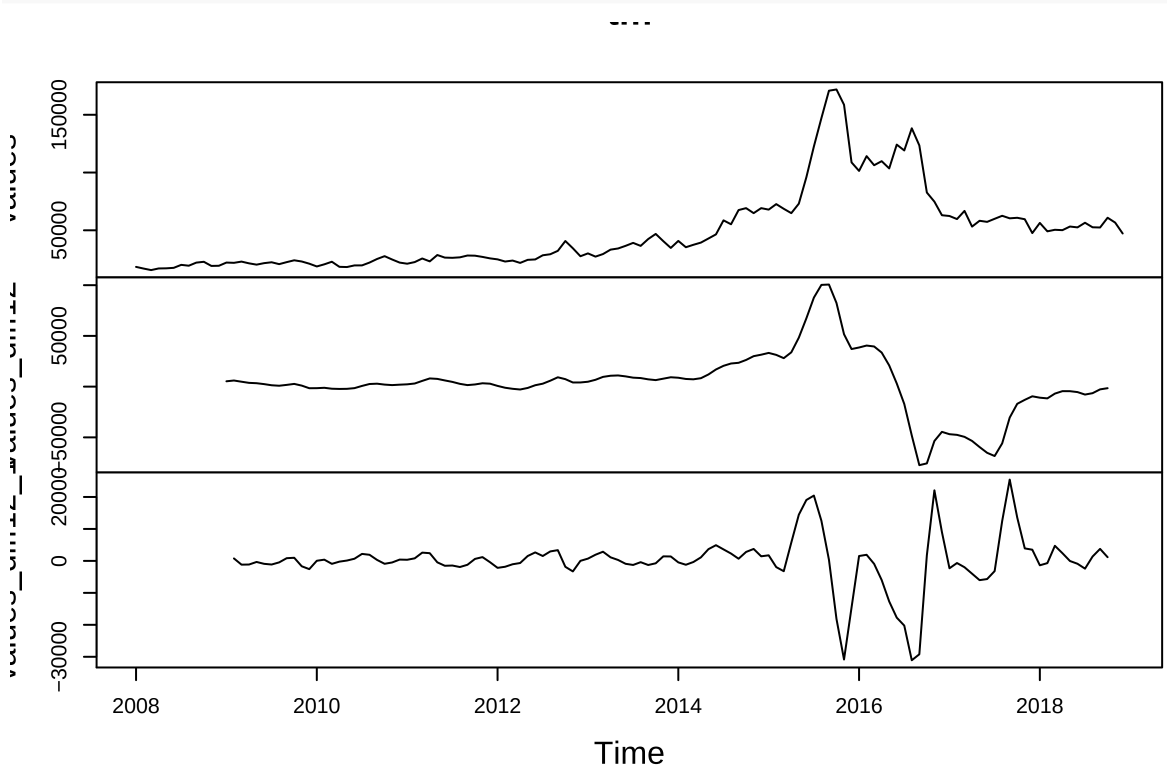
seasonal differencing

```r
values_diff12 = diff(values.fit.ma, lag = 12)
tm <- cbind(values, values_diff12)
plot(tm)
```



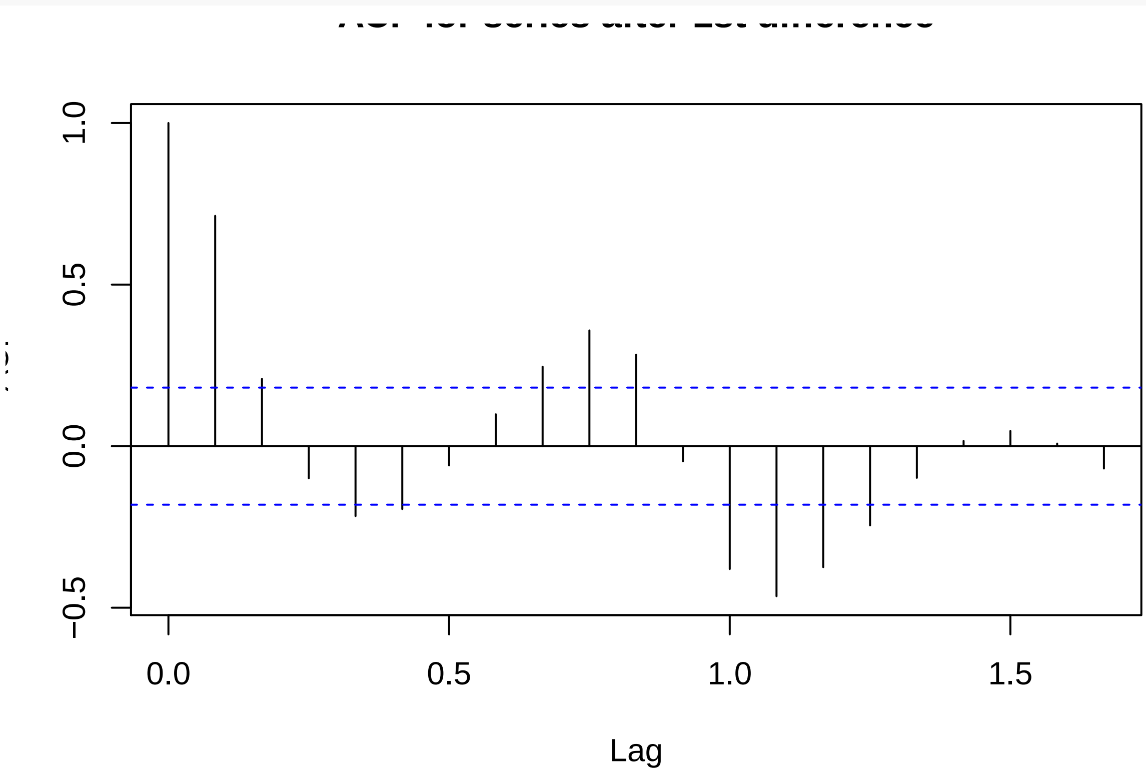
trend remains, take 1st difference

```r
values_diff12_1 = diff(values_diff12)
tm <- cbind(values, values_diff12, values_diff12_1)
plot(tm)
```



A first difference seems to suffice here, but let's check again to confirm the visual inspection by using an ACF plot and ADF test:

```r
acf(na.omit(values_diff12_1), main="ACF for series after 1st difference")
```

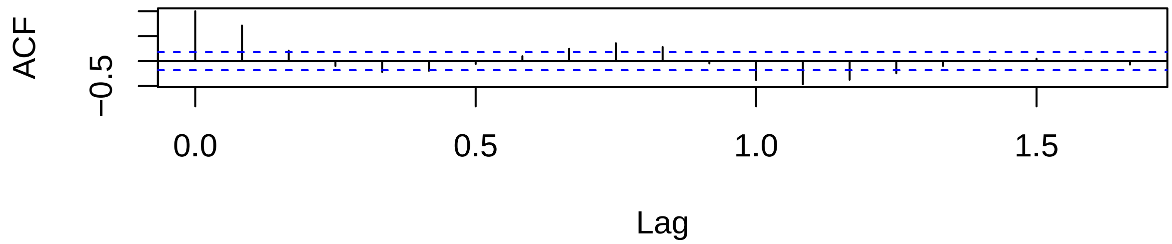


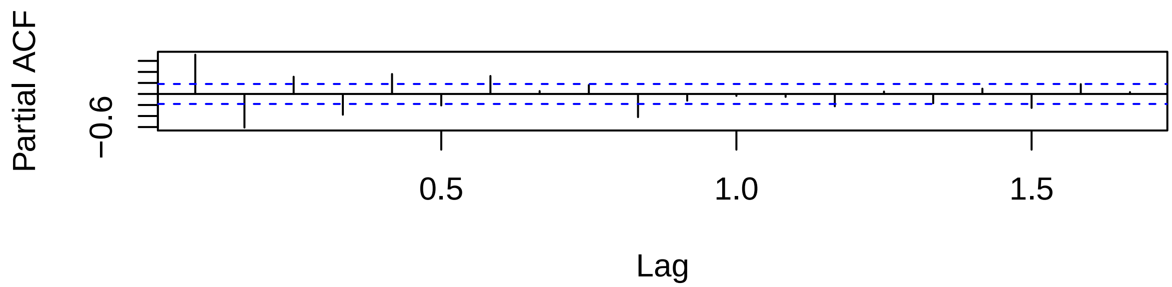

```r
adf.test(na.omit(values_diff12_1))
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  na.omit(values_diff12_1)
## Dickey-Fuller = -3.5973, Lag order = 4, p-value = 0.03656
## alternative hypothesis: stationary
```

Both the ACF plot and ADF test confirm non-stationarity after the first difference. There's a sort of pattern going on on this ACF plot, probably due to the seasonal component - not a pure SMA process, indicates AR.

```r
diff.values <- na.omit(values_diff12_1)
par(mfrow=c(2,1))
acf(diff.values, main="")
pacf(diff.values, main="")
```





A relatively slow decay until lag 6, might indicate a AR of order 6. Negative spike at lag 1 is possible sAR(1) term, it repeats.

---

5. Model

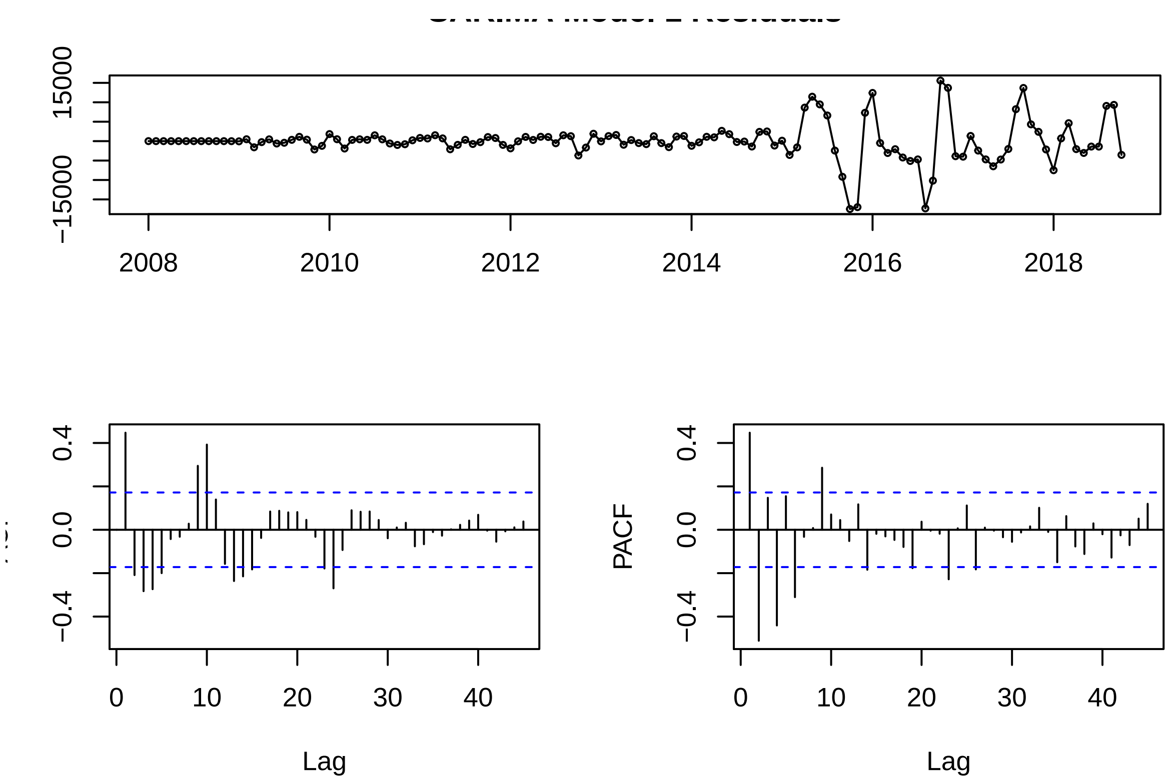
SARIMA(p, d, q)(P, D, Q) order

Try for up to 1 SAR, 1 difference, 1 seasonal D, 1-6 AR, review ACF and PACF if no good fits found.

```
#start with SAR(1)
values.fit.sarima1 <- arima(values.fit.ma, order=c(1, 1, 0), seasonal = list(order = c(1, 1, 0), period
values.fit.sarima1
```

```
##
## Call:
## arima(x = values.fit.ma, order = c(1, 1, 0), seasonal = list(order = c(1, 1,
##     0), period = 12))
##
## Coefficients:
##          ar1     sar1
##       0.7262  -0.4014
## s.e.  0.0630   0.0797
##
## sigma^2 estimated as 25281493:  log likelihood = -1164.6,  aic = 2335.21
```

```
#residuals
tsdisplay(residuals(values.fit.sarima1), lag.max=45, main='SARIMA Model 1 Residuals')
```



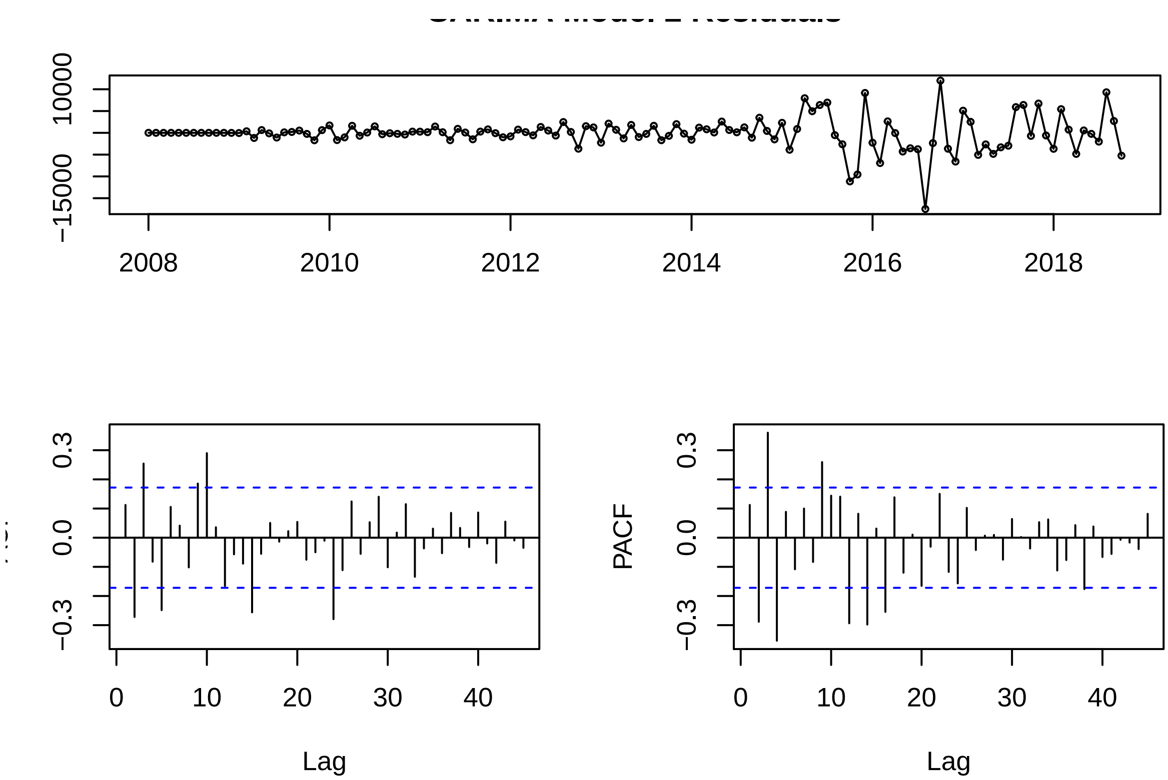

```r
#forecast accuracy
forecast.sarima1 <- forecast(values.fit.sarima1, h=12) #12months
accuracy(forecast.sarima1)
```

```
##                    ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 4.127787 4770.048 2765.512 0.2906328 4.655959 0.1456898
##                   ACF1
## Training set 0.4474625
```

```r
#1 ar wasnt good, look at 3
values.fit.sarima2 <- arima(values.fit.ma, order=c(3, 1, 0), seasonal = list(order = c(1, 1, 0), period
values.fit.sarima2
```

```
##
## Call:
## arima(x = values.fit.ma, order = c(3, 1, 0), seasonal = list(order = c(1, 1,
##     0), period = 12))
##
## Coefficients:
##          ar1      ar2     ar3     sar1
##       1.3649  -0.9807  0.2998  -0.4206
## s.e.  0.0895   0.1283  0.0898   0.0816
##
## sigma^2 estimated as 14146123:  log likelihood = -1131.38,  aic = 2272.76
```

```r
#residuals
tsdisplay(residuals(values.fit.sarima2), lag.max=45, main='SARIMA Model 2 Residuals')
```




```r
#forecast accuracy
forecast.sarima2 <- forecast(values.fit.sarima2, h=12) #12months
accuracy(forecast.sarima2)
```

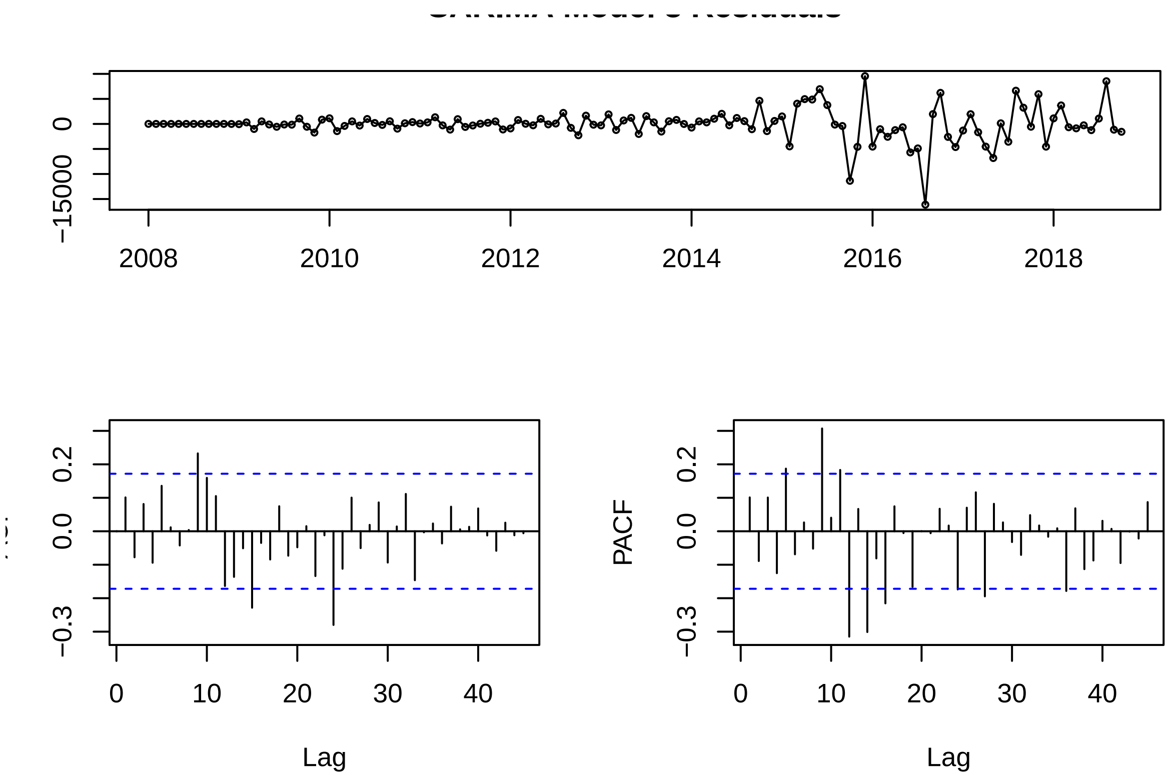
```
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -58.74926 3568.127 2171.997 0.1388934 3.892016 0.1144229
##                   ACF1
## Training set 0.1122667
```

Results aren't very good

```r
#ar6
values.fit.sarima3 <- arima(values.fit.ma, order=c(6, 1, 0), seasonal = list(order = c(1, 1, 0), period
values.fit.sarima3
```

```
##
## Call:
## arima(x = values.fit.ma, order = c(6, 1, 0), seasonal = list(order = c(1, 1,
##     0), period = 12))
##
## Coefficients:
##          ar1      ar2     ar3      ar4     ar5      ar6     sar1
##       1.6753  -1.8012  1.5463  -1.2726  0.7647  -0.2654  -0.4099
## s.e.  0.0894   0.1658  0.2026   0.2037  0.1685   0.0915   0.0818
##
## sigma^2 estimated as 10144043:  log likelihood = -1112.64,  aic = 2241.27
```

```r
#residuals
tsdisplay(residuals(values.fit.sarima3), lag.max=45, main='SARIMA Model 3 Residuals')
```




```
#forecast accuracy
forecast.sarima3 <- forecast(values.fit.sarima3, h=12) #12months
accuracy(forecast.sarima3)
```

```
##                     ME     RMSE      MAE       MPE     MAPE       MASE
## Training set -61.5572 3021.532 1750.562 0.1144758 3.102899 0.09222128
##                   ACF1
## Training set 0.1011675
```

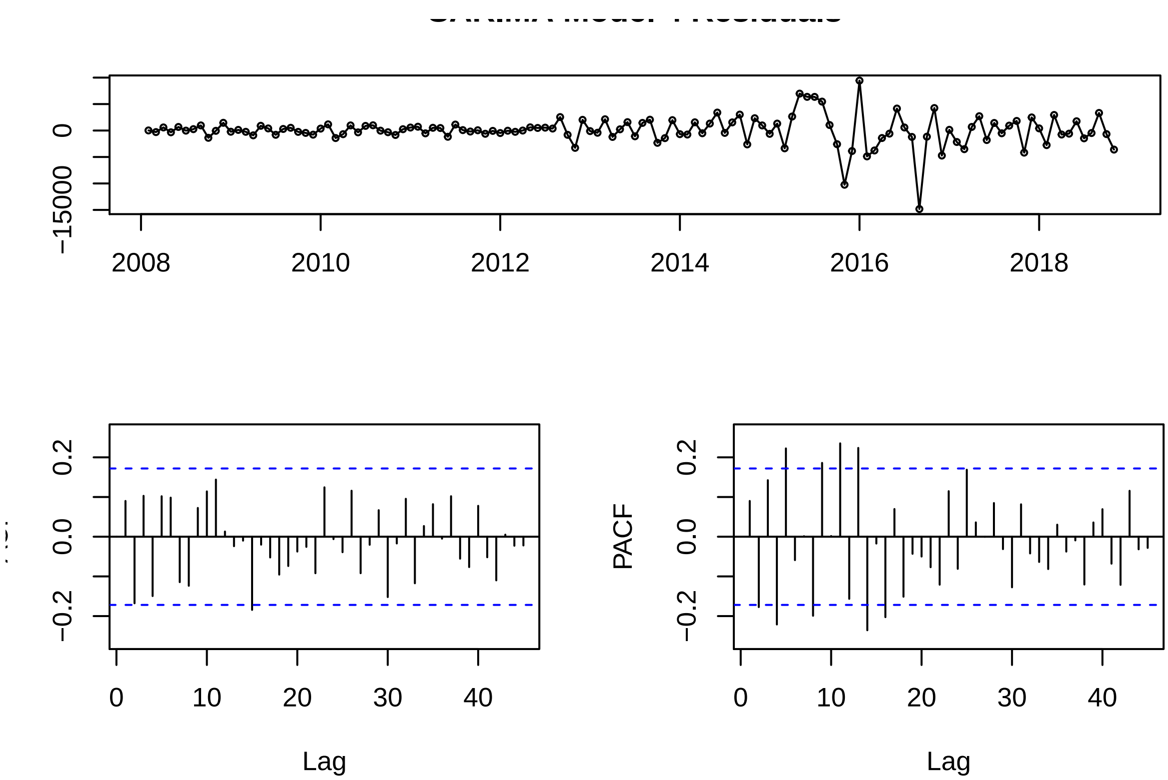
Auto.arima

Another method is to let auto.arima estimate parameters and see how they fit with our conclusions so far:

```
values.fit.sarima4 <- auto.arima(ma.fit, seasonal = TRUE)
values.fit.sarima4
```

```
## Series: ma.fit
## ARIMA(5,1,0)(1,0,0)[12]
##
## Coefficients:
##          ar1      ar2     ar3      ar4     ar5    sar1
##       1.6199  -1.6228  1.2198  -0.7996  0.2690  0.1627
## s.e.  0.0855   0.1512  0.1786   0.1520  0.0862  0.0898
##
## sigma^2 estimated as 7622202:  log likelihood=-1203.79
## AIC=2421.57   AICc=2422.5   BIC=2441.59
```

```
#residuals
tsdisplay(residuals(values.fit.sarima4), lag.max=45, main='SARIMA Model 4 Residuals')
```



So, our initial interpretation of PACF was wrong. . . .auto suggests 5 non-seasonal terms and 1 seasonal for AR. . Let's keep this auto model and check for other types of models

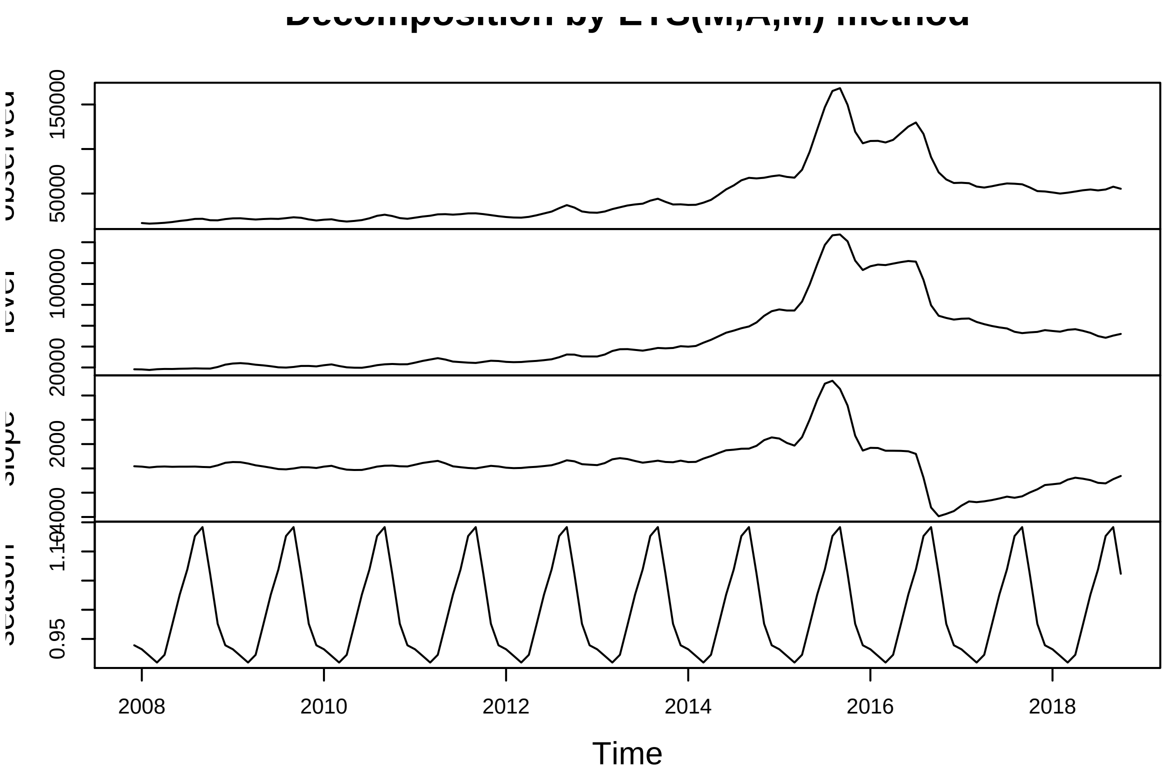
Holt Winters

Trying a different model, exponential smoothing

```
values.fit.hw1 <- ets(values.fit.ma, model="MAM", damped=FALSE)
values.fit.hw1
```

```
## ETS(M,A,M)
##
## Call:
##  ets(y = values.fit.ma, model = "MAM", damped = FALSE)
##
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 0.1047
##     gamma = 1e-04
##
##   Initial states:
##     l = 18282.9878
##     b = 176.2215
##     s = 0.9391 0.9758 1.0617 1.1418 1.1265 1.0693
##            1.0262 0.9741 0.9228 0.9095 0.921 0.9322
##
##   sigma:  0.0568
##
##      AIC     AICc      BIC
## 2656.877 2662.341 2705.625
```

```r
plot(values.fit.hw1)
```




```r
accuracy(values.fit.hw1)
```

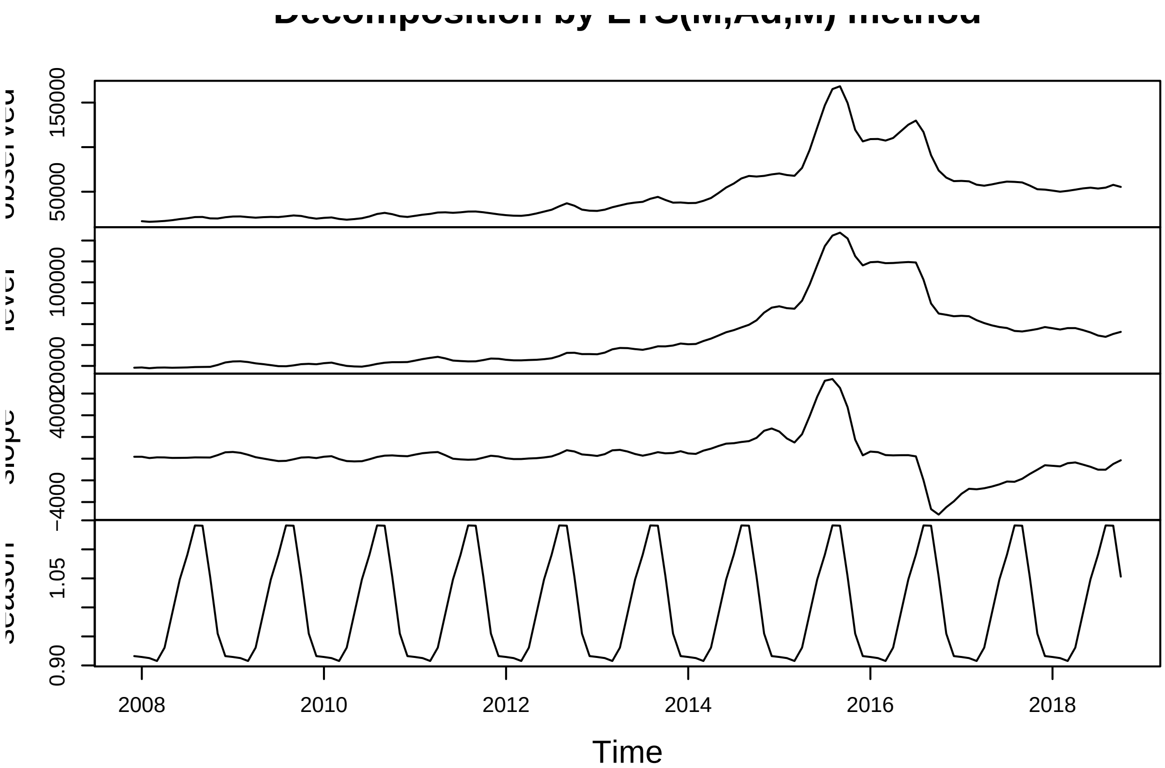
```
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -105.8016 4836.084 2297.052 0.2373048 3.912748 0.1210109
##                   ACF1
## Training set 0.6923798
```

```r
values.fit.hw2 <- ets(values.fit.ma, model="MAM",
                      damped=TRUE)
values.fit.hw2
```

```
## ETS(M,Ad,M)
##
## Call:
##  ets(y = values.fit.ma, model = "MAM", damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.9998
##     beta  = 0.1311
##     gamma = 2e-04
##     phi   = 0.9647
##
##   Initial states:
##     l = 18282.7484
##     b = 176.2974
##     s = 0.9161 0.9548 1.053 1.1408 1.1412 1.0908
##            1.0482 0.9895 0.9307 0.9077 0.9127 0.9146
##
```

```
##   sigma:  0.0561
##
##      AIC     AICc      BIC
## 2654.275 2660.437 2705.890
```

```r
plot(values.fit.hw2)
```




```r
accuracy(values.fit.hw2)
```

```
##                   ME     RMSE      MAE       MPE     MAPE     MASE
## Training set 26.0826 4468.495 2220.006 0.3683582 3.867666 0.116952
##                   ACF1
## Training set 0.6704183
```

pick the model with damped trend, hw2, marginally better

---

6. Model evaluation and forecasting

Accuracy

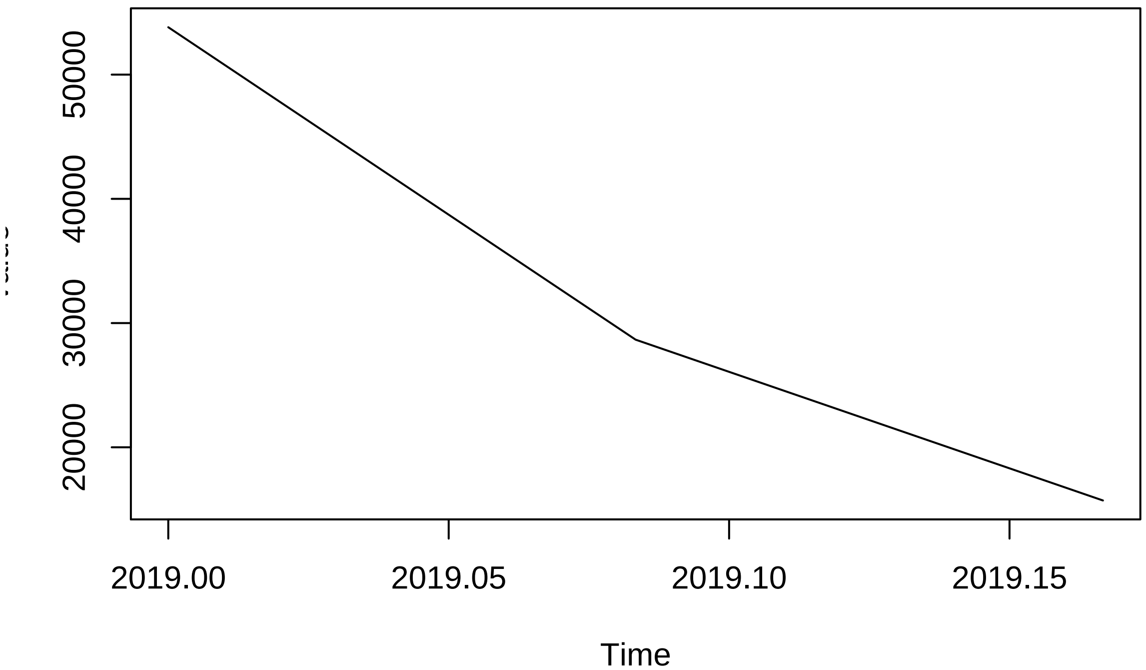
Compare with real observed values for the first three months of 2019

```r
data2019 <- read_csv("migr_asylum_applicant_EU28/migr_asyappctzm_2019_Data.csv")
```

```
## Parsed with column specification:
## cols(
##   TIME = col_character(),
##   GEO = col_character(),
##   CITIZEN = col_character(),
##   SEX = col_character(),
```

```
##   AGE = col_character(),
##   ASYL_APP = col_character(),
##   UNIT = col_character(),
##   Value = col_number(),
##   `Flag and Footnotes` = col_character()
## )
```

```
values2019 <- ts(data2019[8], start = c(2019, 1), frequency=12)
plot.ts(values2019)
```



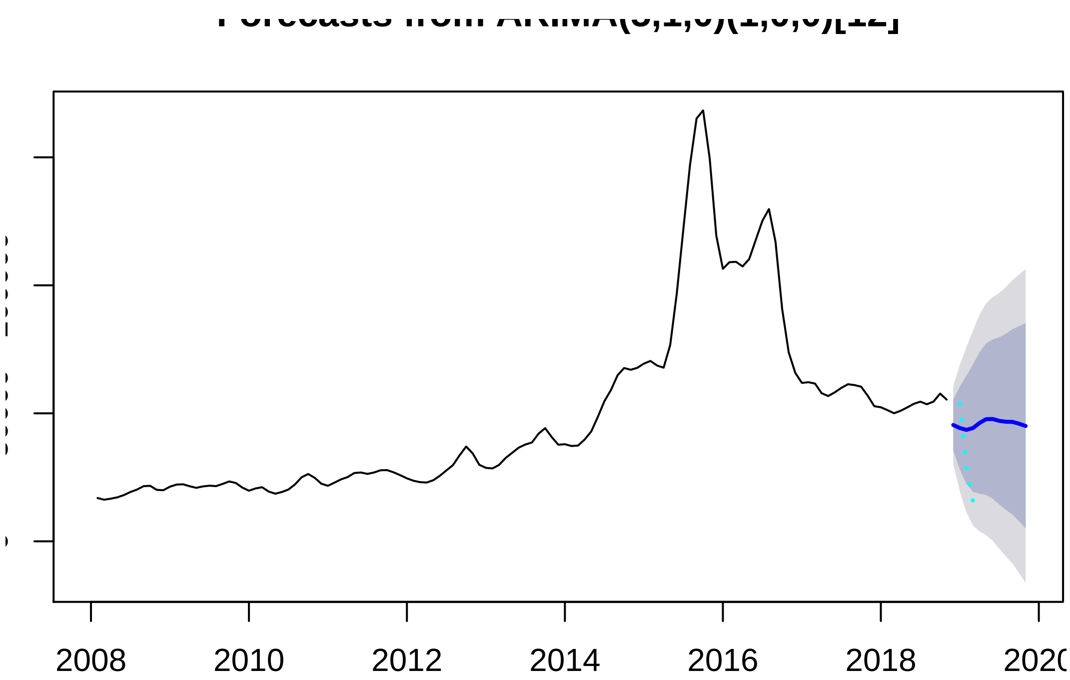
A strong downward trend

Forecasting using the chosen model(s)

Forecasting the next 12 months

Seasonal ARIMA

```
forecast.values.sarima4 <- forecast(values.fit.sarima4, h=12)
plot(forecast.values.sarima4)
lines(values2019, col="cyan", lwd=2, lty=3)
```



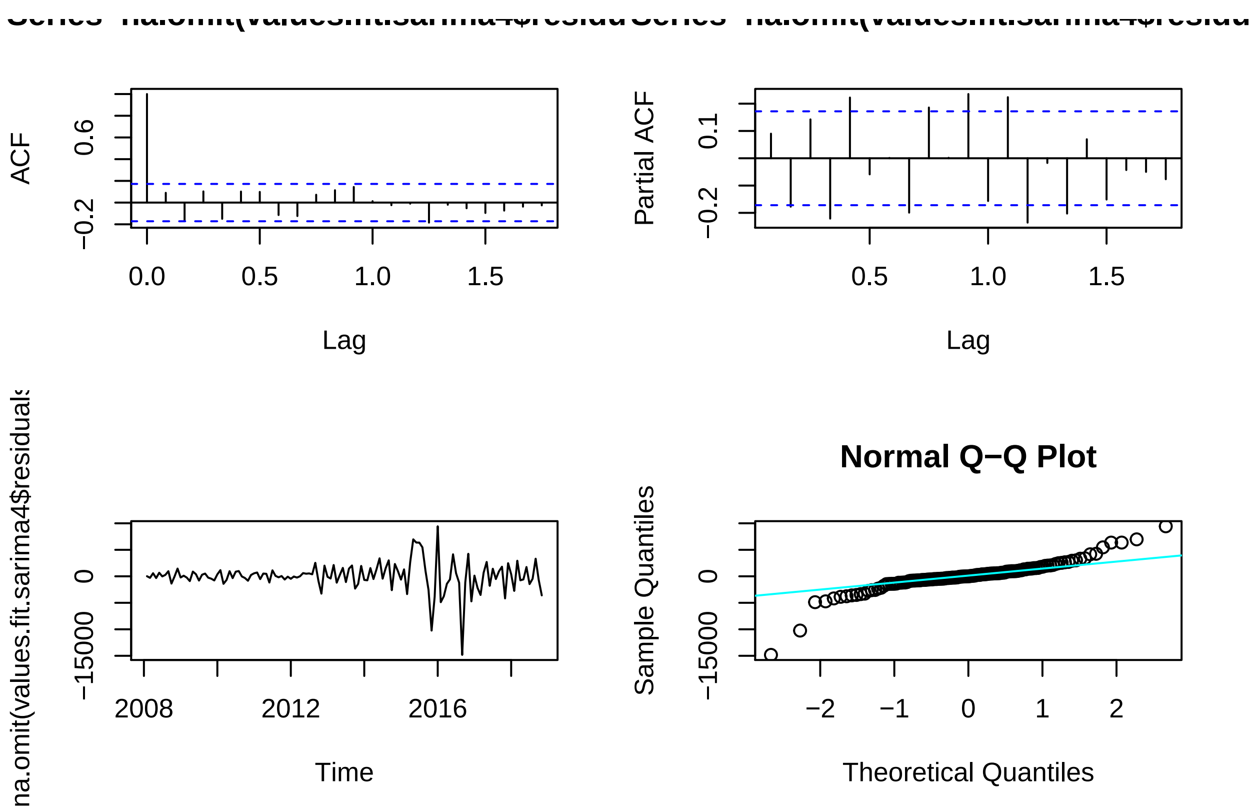

```r
accuracy(forecast.values.sarima4, values2019)
```

```
##                        ME      RMSE       MAE         MPE      MAPE
## Training set     56.51262  2685.475  1659.178   0.4042005  3.198991
## Test set     -11288.61921 19383.377 17658.847 -71.8985574 83.735829
##                    MASE       ACF1 Theil's U
## Training set 0.08740708  0.0901048        NA
## Test set     0.93028488 -0.0173775  1.589794
```

```r
#test residuals
Box.test(residuals(values.fit.sarima4), type="Ljung-Box") #test alternative hypothesis that there is no
```

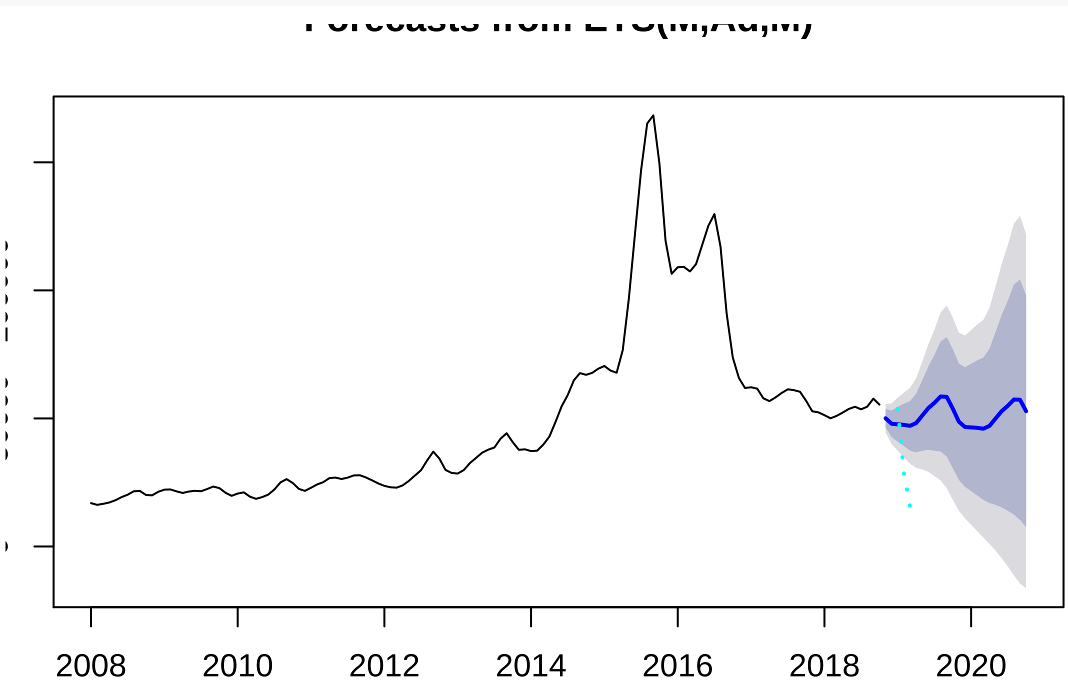
```
##
##  Box-Ljung test
##
## data:  residuals(values.fit.sarima4)
## X-squared = 1.08, df = 1, p-value = 0.2987
```

```r
par(mfrow=c(2,2))
acf(na.omit(values.fit.sarima4$residuals))
pacf(na.omit(values.fit.sarima4$residuals))
plot(na.omit(values.fit.sarima4$residuals))
qqnorm(na.omit(values.fit.sarima4$residuals))
qqline(na.omit(values.fit.sarima4$residuals), col="cyan")
```



Holt Winters seasonal smoothing

```r
plot(forecast(values.fit.hw2))
lines(values2019, col="cyan", lwd=2, lty=3)
```



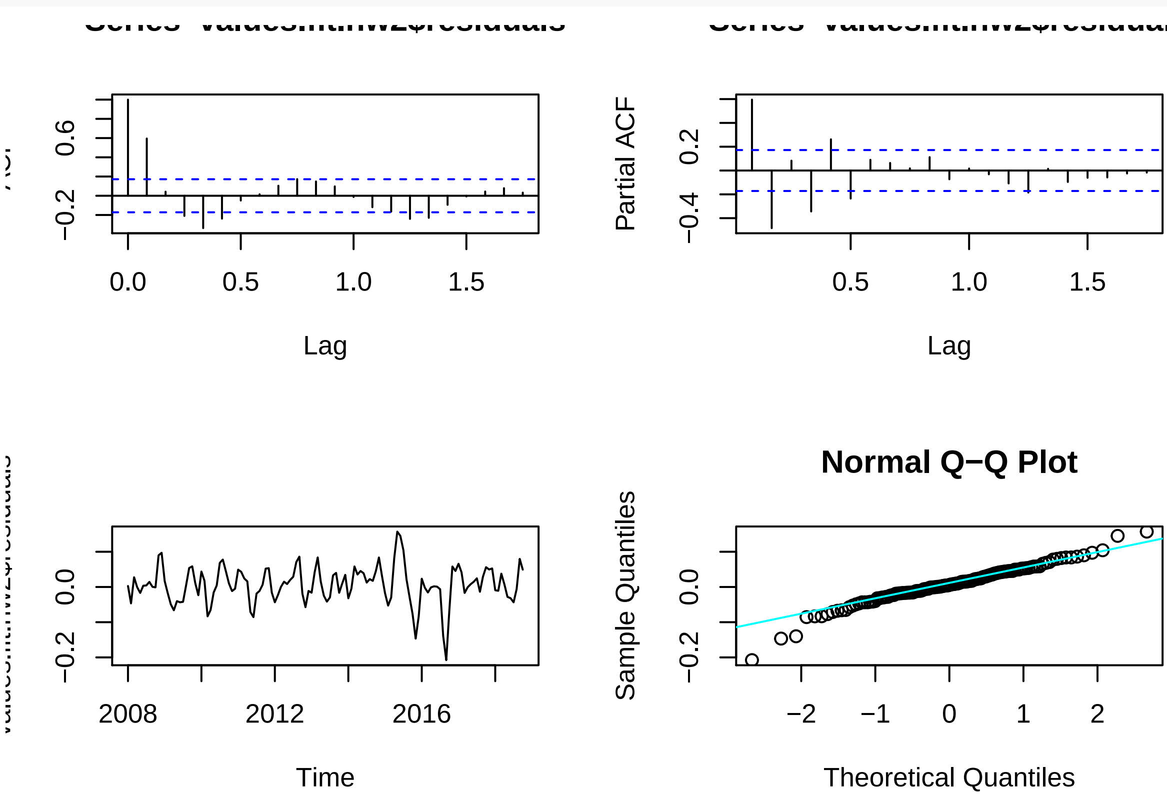
```r
accuracy(forecast(values.fit.hw2), values2019)
```

```
##                        ME      RMSE       MAE         MPE      MAPE
## Training set     26.0826  4468.495  2220.006   0.3683582  3.867666
## Test set     -14691.8277 21416.809 18769.668 -84.6273976 92.204913
##                  MASE        ACF1 Theil's U
## Training set 0.116952  0.67041833        NA
## Test set     0.988804 -0.02329674  1.768313
```

```r
#test residuals
Box.test(residuals(values.fit.hw2), type="Ljung-Box") #test alternative hypothesis that there is non-ze
```

```
##
##  Box-Ljung test
##
## data:  residuals(values.fit.hw2)
## X-squared = 47.177, df = 1, p-value = 6.485e-12
```

```r
par(mfrow=c(2,2))
acf(values.fit.hw2$residuals)
pacf(values.fit.hw2$residuals)
plot(values.fit.hw2$residuals)
qqnorm(values.fit.hw2$residuals)
qqline(values.fit.hw2$residuals, col="cyan")
```



not great on the residuals correlations accuracy is bad on this test

```r
forecast.values.sarima4
```

```
##          Point Forecast     Lo 80    Hi 80      Lo 95    Hi 95
## Dec 2018       45451.82 35529.760 55373.87 30277.3405 60626.29
## Jan 2019       44259.66 28054.117 60465.20 19475.4218 69043.89
```

```
## Feb 2019       43551.34 22416.156 64686.52  11227.8662  75874.81
## Mar 2019       44254.86 19334.940 69174.78   6143.1303  82366.59
## Apr 2019       46249.75 18557.065 73942.43   3897.4433  88602.06
## May 2019       47719.25 18110.435 77328.07   2436.4733  93002.04
## Jun 2019       47771.61 16640.342 78902.87    160.4459  95382.77
## Jul 2019       47060.43 14381.062 79739.80  -2918.3507  97039.21
## Aug 2019       46712.42 12322.702 81102.14  -5882.1142  99306.95
## Sep 2019       46659.28 10441.561 82877.00  -8730.9412 102049.50
## Oct 2019       45938.72  7842.294 84035.15 -12324.7356 104202.18
## Nov 2019       45058.51  4956.470 85160.55 -16272.2678 106389.29
```