Time series analysis: Seasonal adjustment and forecasting

Ozan Bakış[1]

[1]Bahcesehir University, Department of Economics and BETAM

Outline

Seasonal adjustment I

TS components:

- **trend-cycle:** ”Trend” is long-term direction while ”cycle” refers to deviation from trend due to other factors that are not of fixed period. Length of the current cycle is unknown. Standard models (of exponential smoothing) allow for seasonality (not cyclicity). This is why cycle component is “within” trend component in the standard TS decompositions. Under certain conditions ARIMA models allow for both cyclic and seasonal behavior, but we will follow the standard approach.

- **seasonal:** deviation from trend due to seasonal factors. Seasonality is always of a fixed and known period. Periodicity in hours of the day, days of the week, weeks of the month, months/quarters of the year...

- **remainder:** Residual part.

Seasonal adjustment II

- There are 2 alternative models that can be used for TS decomposition: additive vs. multiplicative:

$$y_t = TC_t + S_t + E_t \quad \text{vs} \quad y_t = TC_t \times S_t \times E_t$$

where TC: trend-cycle, S: seasonal and E: remainder.

- **Seasonally adjusted data:** By removing the seasonal part from the original TS data we get seasonally adjusted data, i.e. $y_t - S_t$ or y_t/S_t according to the model used for decomposition.

- Since seasonally adjusted data contain both "trend-cyle" and "remainder" components, they are not smooth and they may show high variability. For a change in the structure of the economy one should look at the trend-cycle component rather than seasonally adjusted data. For the remaining slides, we will use "trend" as short form of "trend-cycle".

Seasonal adjustment III

- **Remark:** Weekly and daily data need some special care because of the underlying irregular pattern in the data.
- Let us begin by plotting monthly electricity consumption in Turkey from Jan 2010 and December 2017.

```
url = "https://github.com/obakis/econ_data/raw/master/elektrik_df.rds"
download.file(url, "elektrik_df.rds", mode ="wb")
eldf = readRDS("elektrik_df.rds")
```

- Create a `ts` object using electricity consumption data and plot it

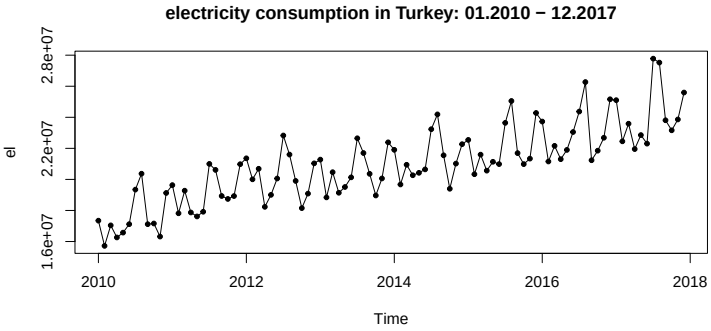
```
head(eldf,3)
```

```
##   year month       el
## 1 2010     1 17343871
## 2 2010     2 15720451
## 3 2010     3 17041086
```

```
el = ts(eldf$el, start = 2010, frequency = 12) # el is a ts object now!

plot(el, main="electricity consumption in Turkey: 01.2010 - 12.2017",
     type="o",pch = 20) # type="b"
```

Seasonal adjustment IV



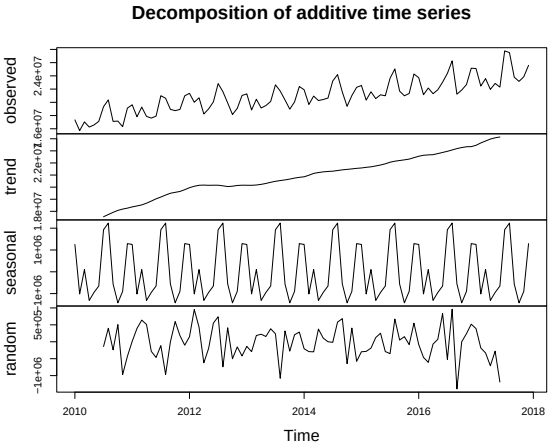
- By looking at the plot, we see that there is not only a trend but also a clear seasonal pattern.

Seasonal adjustment V

- But, how to decompose the original series into ”trend”, ”seasonal” and ”residual” components?
- There are two functions for TS decomposition in base R: **`decompose()`** and **`stl()`**. The former provides basic functionality using MAs while the latter is more sophisticated and relies on loess smoothing (STL is an acronym for “Seasonal and Trend decomposition using Loess”).
- Both of these decomposition methods allow us to compute both ”seasonal effects” and ”seasonally adjusted” series.
- The basic one is **`decompose()`** function.

```
el_dec = decompose(el, "additive")
plot(el_dec)
```

Seasonal adjustment VI



Seasonal adjustment VII

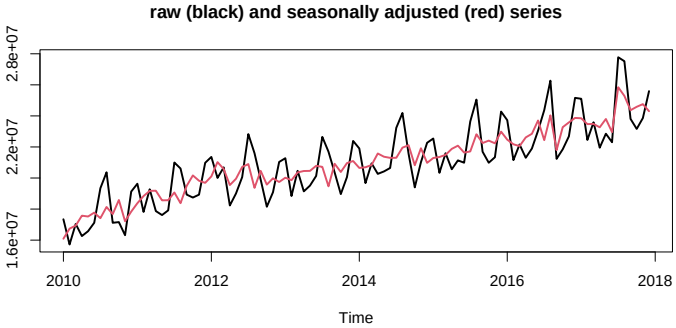
- Seasonally adjusted data can be obtained by subtracting the estimated seasonal component from the original data.

```
names(el_dec)
```

```
## [1] "x"        "seasonal" "trend"    "random"   "figure"   "type"
```

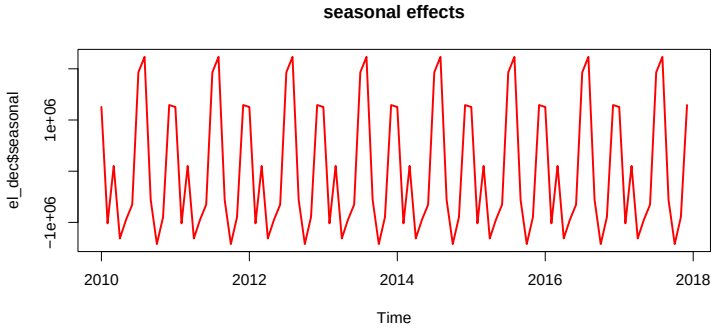
```
el_sa = el - el_dec$seasonal
# ts.plot for plotting multiple series
ts.plot(el, el_sa, lty=1, col=1:2, lwd=2,
        main="raw (black) and seasonally adjusted (red) series")
```

Seasonal adjustment VIII



Seasonal adjustment IX

```
ts.plot(el_dec$seasonal, lty=1, col="red", lwd=2, main="seasonal effects")
```



Seasonal adjustment X

- But what is going on behind the scenes? To understand it, let us try to get seasonal effects manually. For this we will use again the OLS regression wehere we computed a liner trend. In reality, the underlying trend is almost never linear. But, this illustrates the idea.

```
tail(eldf,3)
```

```
##    year month       el
## 94 2017    10 23161713
## 95 2017    11 23860824
## 96 2017    12 25594468
```

```
eldf$tr = 1:nrow(eldf)
head(eldf,3)
```

```
##   year month       el tr
## 1 2010     1 17343871  1
## 2 2010     2 15720451  2
## 3 2010     3 17041086  3
```

```
reg_t=lm(el ~ tr, data=eldf)
coef(summary(reg_t))
```

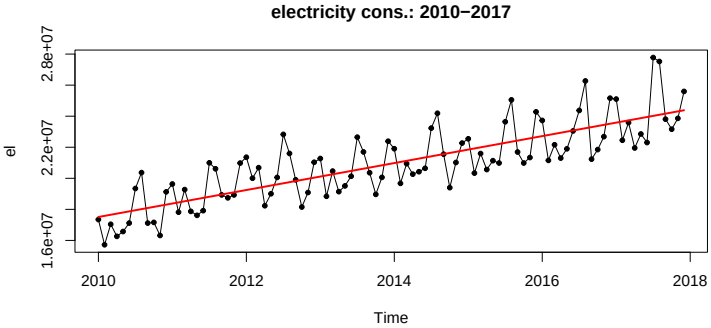
Seasonal adjustment XI

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 17436421     296718    58.8 6.49e-76
## tr             72345       5312    13.6 5.75e-24
```

```
### get fitted values and convert them into "ts" object
#el = ts(eldf$el, start = 2010, frequency = 12) # el is already an ts object
fit_el = ts(fitted(reg_t), start=2010, freq=12)
```

Seasonal adjustment XII

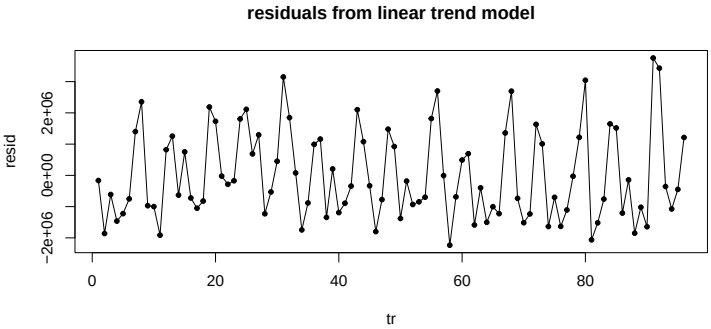
```
plot(el, main="electricity cons.: 2010-2017", type="o",pch = 20)
lines(fit_el, col="red", lwd=2)
```



Seasonal adjustment XIII

- First, we will get residuals (detrended data) from this regression.

```
eldf$resid = residuals(reg_t)
plot(resid ~ tr, data=eldf, main="residuals from linear trend model",
     type="o",pch = 20)
```



Seasonal adjustment XIV

- Then, we will find the mean value of ”residuals” for each month (optionally, we will normalize these mean values so that their mean is zero).

- For computing mean values for each month we will use again a simple regression

```
#aggregate(resid ~ month, FUN=mean, data=eldf)
reg_r = lm(resid ~ factor(month), data=eldf)
summary(reg_r)$coef
```

```
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      1063345     227437   4.675 1.11e-05
## factor(month)2  -2183121     321645  -6.787 1.51e-09
## factor(month)3  -1036659     321645  -3.223 1.81e-03
## factor(month)4  -2378851     321645  -7.396 9.68e-11
## factor(month)5  -2022620     321645  -6.288 1.38e-08
## factor(month)6  -1695929     321645  -5.273 1.03e-06
## factor(month)7   1060198     321645   3.296 1.44e-03
## factor(month)8   1296455     321645   4.031 1.22e-04
## factor(month)9  -1614827     321645  -5.021 2.85e-06
```

Seasonal adjustment XV

```
## factor(month)10 -2460066     321645  -7.648 3.05e-11
## factor(month)11 -1922211     321645  -5.976 5.34e-08
## factor(month)12   197497     321645   0.614 5.41e-01
```

```
eldf$mon_ef = fitted(reg_r)
#eldf$mon_ef = eldf$mon_ef - mean(eldf$mon_ef) # optional step
head(eldf,16)
```

```
##    year month       el tr    resid   mon_ef
## 1  2010     1 17343871  1  -164894  1063345
## 2  2010     2 15720451  2 -1860660 -1119777
## 3  2010     3 17041086  3  -612369    26686
## 4  2010     4 16262031  4 -1463769 -1315507
## 5  2010     5 16573623  5 -1224522  -959276
## 6  2010     6 17117956  6  -752533  -632584
## 7  2010     7 19343511  7  1400677  2123543
## 8  2010     8 20368204  8  2353025  2359800
## 9  2010     9 17116562  9  -970961  -551483
## 10 2010    10 17161127 10  -998742 -1396721
## 11 2010    11 16318433 11 -1913781  -858867
## 12 2010    12 19126995 12   822437  1260841
## 13 2011     1 19631538 13  1254635  1063345
```

Seasonal adjustment XVI

```
## 14 2011     2 17818086 14  -631162 -1119777
## 15 2011     3 19274205 15   752613    26686
## 16 2011     4 17870242 16  -723695 -1315507
```

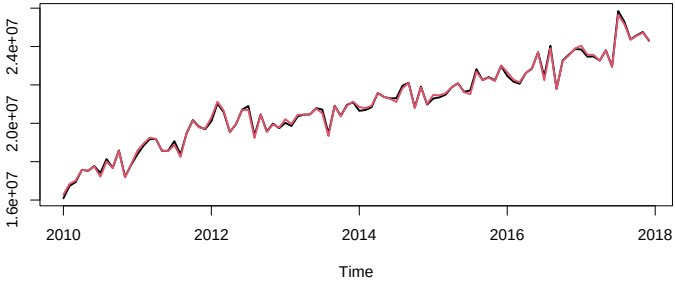
- Subtracting these monthly averages from the original series will yield the seasonally adjusted series.

```
el_sa2 = ts(eldf$el - eldf$mon_ef, start=2010, freq=frequency(el))
```

Seasonal adjustment XVII

- Comparing the "manual way" with the **decompose()** function we see almost no differences

```
ts.plot(el_sa, el_sa2, lty=1, col=1:2, lwd=2)
```



Forecasting with seasonal data I

- TS decomposition can also be used for forecasting. To forecast a decomposed time series, we separately forecast the seasonal component, and the seasonally adjusted trend/cycle component.
- We assume that the seasonal component is unchanging.
- How to make predictions with seasonal data? The idea is to use trend + seasonal effects.
- Let us use data from 2010 to 2016 to predict year 2017 (last 12 months).

```
url = "https://github.com/obakis/econ_data/raw/master/elektrik_df.rds"
download.file(url, "elektrik_df.rds", mode ="wb")
eldf = readRDS("elektrik_df.rds")
head(eldf)

eldf2 = subset(eldf, year != 2017)
eldf2$tr = 1:nrow(eldf2)
tail(eldf2,3)
```

Forecasting with seasonal data II

```
##    year month       el tr
## 82 2016    10 21850083 82
## 83 2016    11 22683005 83
## 84 2016    12 25161281 84
```

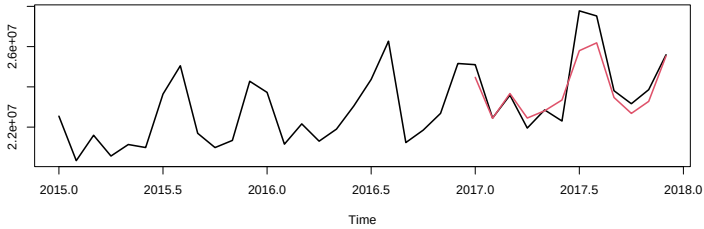
```
reg_t = lm(el ~ tr, data=eldf2)
eldf2$resid = residuals(reg_t)
reg_r = lm(resid ~ factor(month), data=eldf2)

pred_tr = predict(reg_t, newdata = data.frame(tr=85:96))
pred_r = predict(reg_r, newdata = data.frame(month=1:12))

el_fct = ts(pred_tr + pred_r, start = c(2017,1), freq = 12)

el = ts(eldf$el, start = 2010, frequency = 12)
el15 = window(el, start = 2015) # get last 3 years of original data
ts.plot(el15,el_fct, lty=1, col=1:2, lwd=2) # compare fct with observed val.
```

Forecasting with seasonal data III



- The question is how is this successful compared to the famous exponential smoothing methods such as Holt-Winters? Let us check...

```
el2 = window(el, end=c(2016,12)) # use data between 2010 Jan - 2016 Dec.
hw = HoltWinters(el2, gamma=TRUE) # gamma=TRUE for seasonal effects
pred_hw = predict(hw, n.ahead=12) # n.ahead is the # of periods to forecast
el_fct_hw = pred_hw[,"fit"] # "fit" is used to get point estimates
ts.plot(el15,el_fct, el_fct_hw, lty=1, col=1:3, lwd=2)
```

Forecasting with seasonal data IV