Time Series Project

1 Introduction

Monthly hotel occupied room: This time series represents the number of hotel rooms occupied each month over 13 years (1963-1976) in a not specified zone. We can clearly see a growing trend in the data and a defined seasonality whose period appears to be one year. That would make much sense, if we just think about the concept of seasonality for a hotel in a temperate climate zone of the world.

In this project, we applied three filters to smoothing our timeseries: **Moving Average**, **Exponential Smoothing** and **Differencing**.

2 Data description

2.1 Load datas

```
# ts is our time series
ts<-read.csv("monthly-hotel-occupied-room-av-6.csv")
names(ts)[2]="rmNumber" # rename as rmNumber=room number
ts<-na.omit(ts)

# show the example of the data
head(ts,4)
```

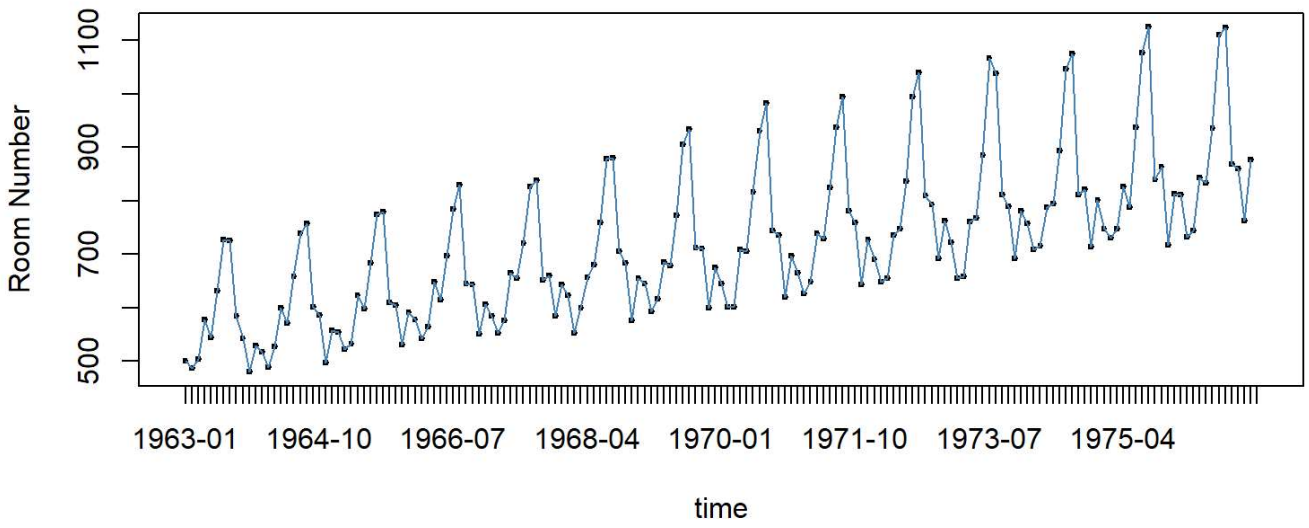
```
##     Month rmNumber
## 1 1963-01      501
## 2 1963-02      488
## 3 1963-03      504
## 4 1963-04      578
```

```
summary(ts)
```

```
##       Month         rmNumber
##  1963-01:   1   Min.    : 480.0
##  1963-02:   1   1st Qu.: 613.5
##  1963-03:   1   Median : 709.5
##  1963-04:   1   Mean    : 722.3
##  1963-05:   1   3rd Qu.: 803.8
##  1963-06:   1   Max.    :1125.0
##  (Other):162
```

2.2 Plot the original data

```
plot(ts, xlab="time", ylab="Room Number", pch=c(16))
lines(ts, type="l", col = "steelblue", xlab="time", ylab="Room Number", pch=c(16))
```



3 Moving Average Smoothing

3.1 Define the moving average function

The moving average filter (SMA) is a common filter used to smoothen a time series and therefore reduce noise. It is particularly useful to identify a possible trend since it reduces fluctuations allowing us to see the general pace of the series.

$$S_t = \frac{y_{t-1}+y_{t-2}+\cdots+y_{t-n}}{n}$$

Where n is the moving window,y_{t-n} is the true time series value, S_t is the expected smoothing result.

In this application we will try to use this filter with different windows and see the outcomes each time.

```
# Moving Average filter function
# n = size of the window of the moving average
# data = time series to apply SMA on
SMA <- function(data, n){
  size=nrow(data)
  result=vector(mode="numeric",length=size)
  sum=0
  a=data$rmNumber
  for (i in 1:n) {
    sum=sum+a[i]
    result[i]=sum/(i+1)
  }
  for (i in (n+1):size) {
    sum = sum-a[i-n]+a[i]
    result[i] = sum/n
  }
  return(result)
}
```

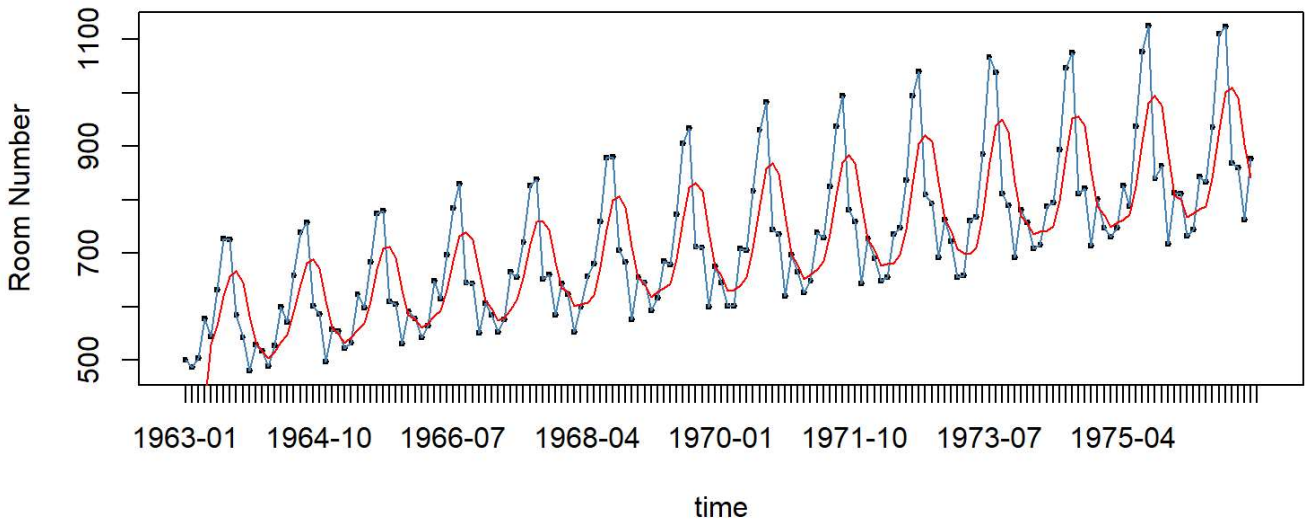
3.2 Moving Average result

In this section we try different window sizes:

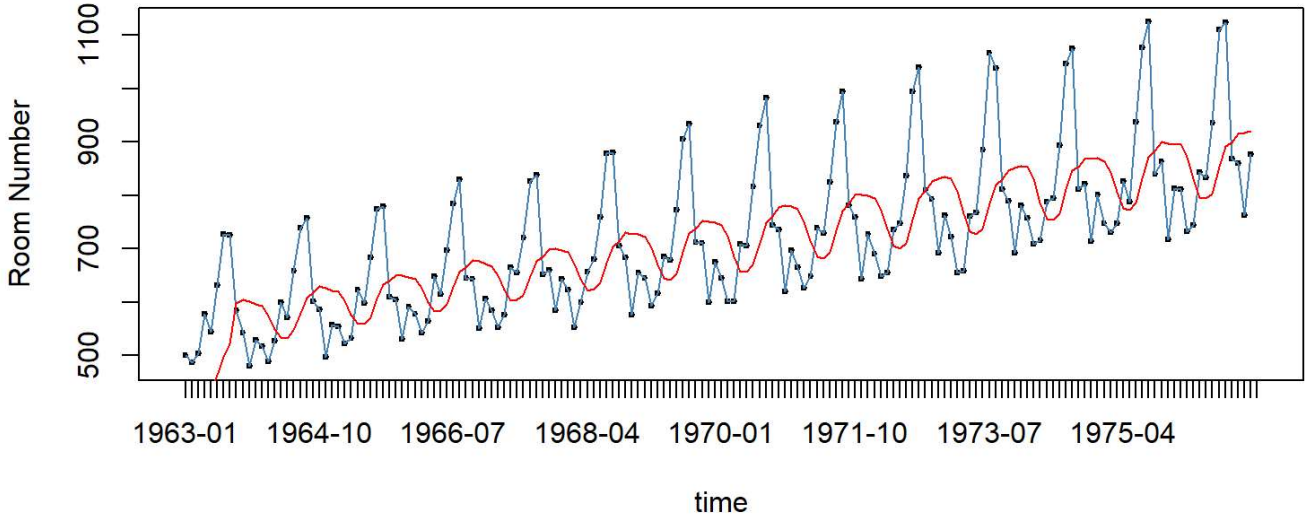
With a small size we obtain smaller reduction in the fluctuations while, on the other hand, with a bigger size we smoothen more our series. Though we can say that such fluctuation are always present unless we choose as window size a multiple of 12 (what we thought being the period of the seasonal component).

- window=4

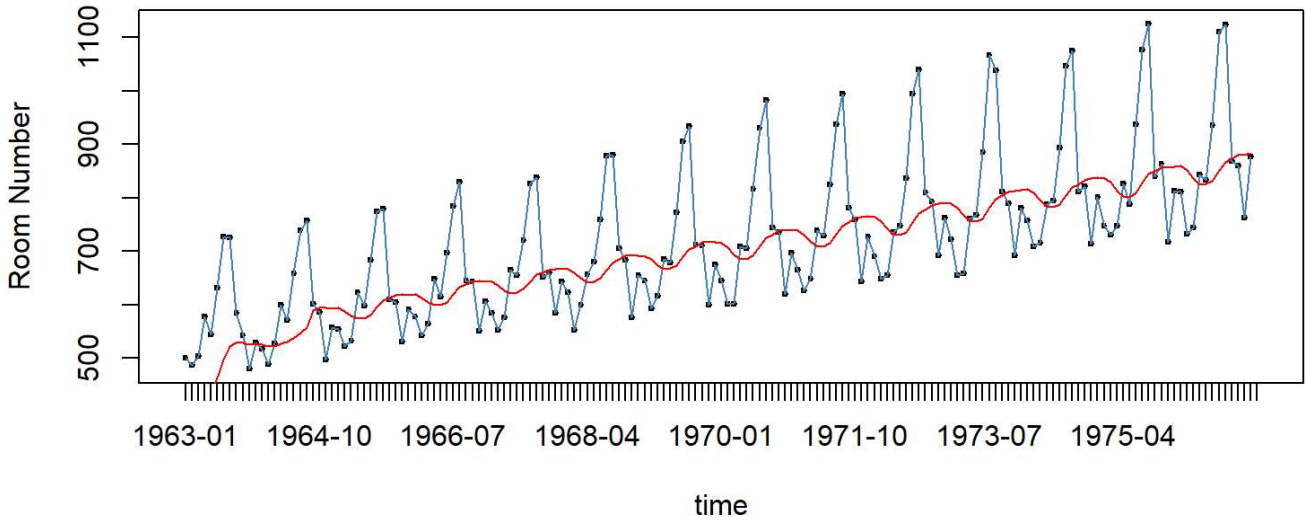
```
window=4 # size of the window of MA
ts1=SMA(ts, window)
plot(ts,xlab="time",ylab="Room Number",pch=c(16))
# plot the lines for the original data
lines(ts,type="l",col = "steelblue",xlab="time",ylab="Room Number",pch=c(16))
# plot the lines for the data after moving average
lines(ts$Month,ts1,type="l",col = "red",xlab="time",ylab="Room Number",pch=c(16))
```



- window=8

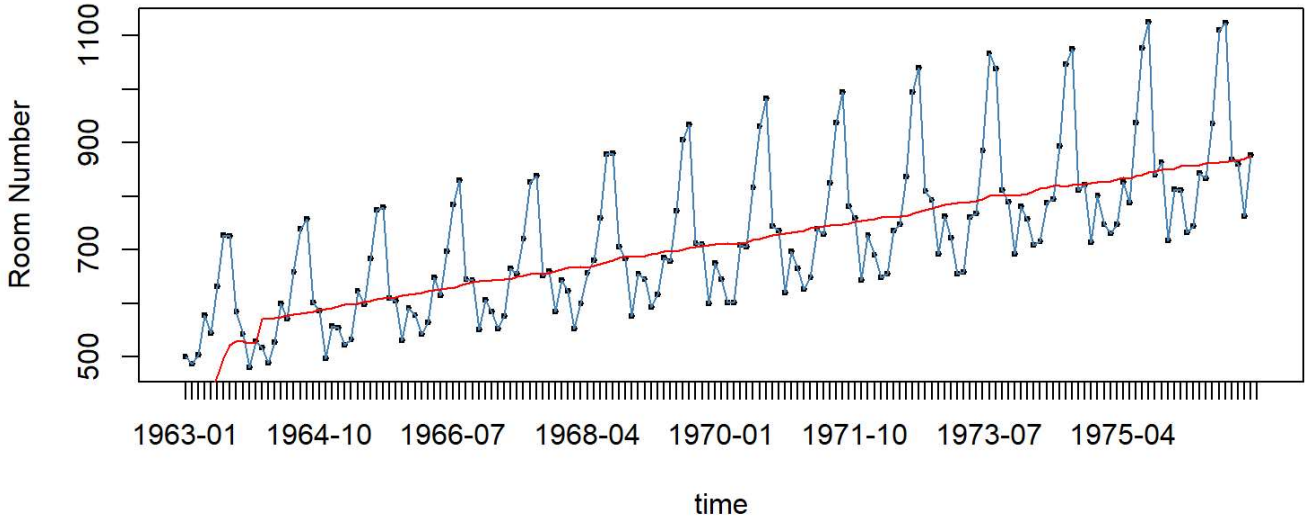


- window=20



As mentioned before, this is the only case where we obtatin a non-oscillating curve, which might be approximated to the trend of the series.

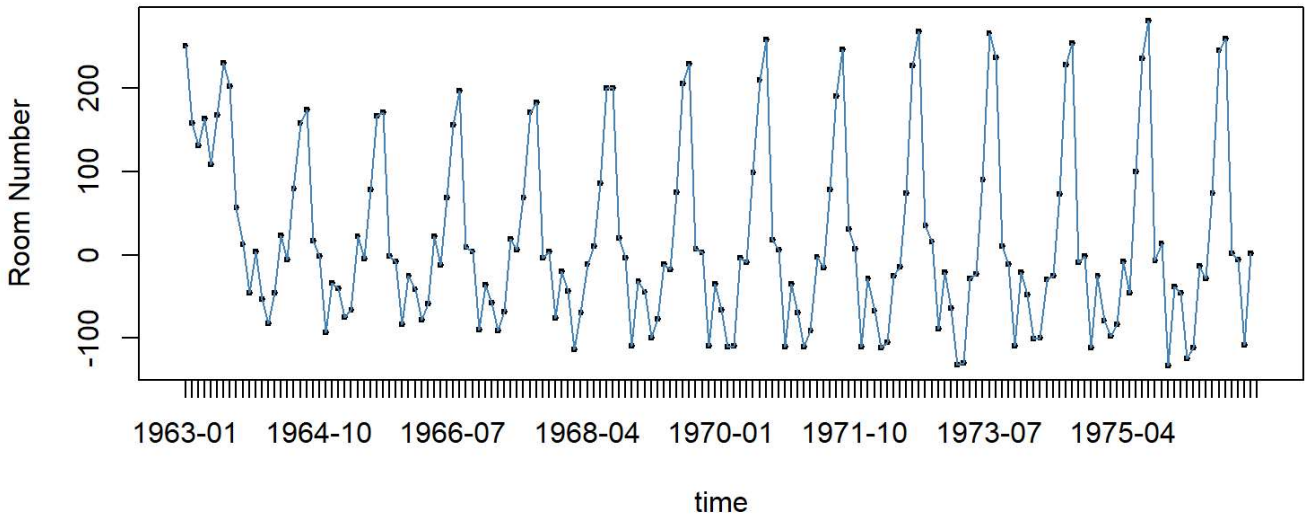
- window=12



3.3 Removing the trend

In this case, we consider the moving average result as the trend of current time series. And we are now trying to remove this component from our original data. Plot to see what the data will be like.

```
ts3=ts$rmNumber-ts1 #ts1 is the result of moving average
plot(ts$Month, ts3, xlab="time", ylab="Room Number", pch=c(16))
lines(ts$Month, ts3, col = "steelblue",  xlab="time", ylab="Room Number", pch=c(16))
```



4 Exponential Smoothing

The third filter that we coded is the exponential smoothing function. We tried also that kind of smoothing to see which one works better compared to SMA. In this case though, the moving average returns results which appear to be way better, especially when we set the size of the filter with the same period of the sesonal component of the series.

We are using this basic exponential smoothing function:

$$S_t = \alpha y_t + (1-\alpha) S_{t-1}$$

Where α is the sommothing coefficient,S_t is the smoothing result in of period t, and y_t is the real value of the time series.

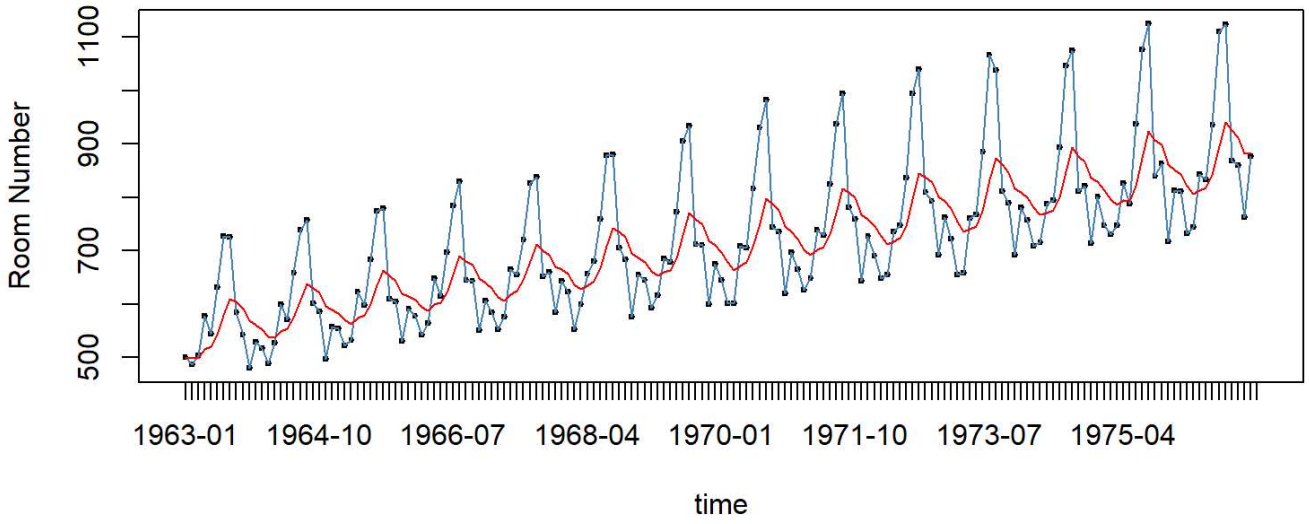
4.1 Define Exponential smoothing function

```
# ES -> exponential_smoothing
# alpha -> smoothing coefficient
# data -> time series to apply ES on
ES <- function(s,alpha){
  size=nrow(s)
  s_out=vector(mode="numeric",length=size)
  s_out[1]=s$rmNumber[1]
  for (i in 2:size) {
    s_out[i]=alpha*s$rmNumber[i]+(1-alpha)*s_out[i-1]
  }
  return(s_out)
}
```

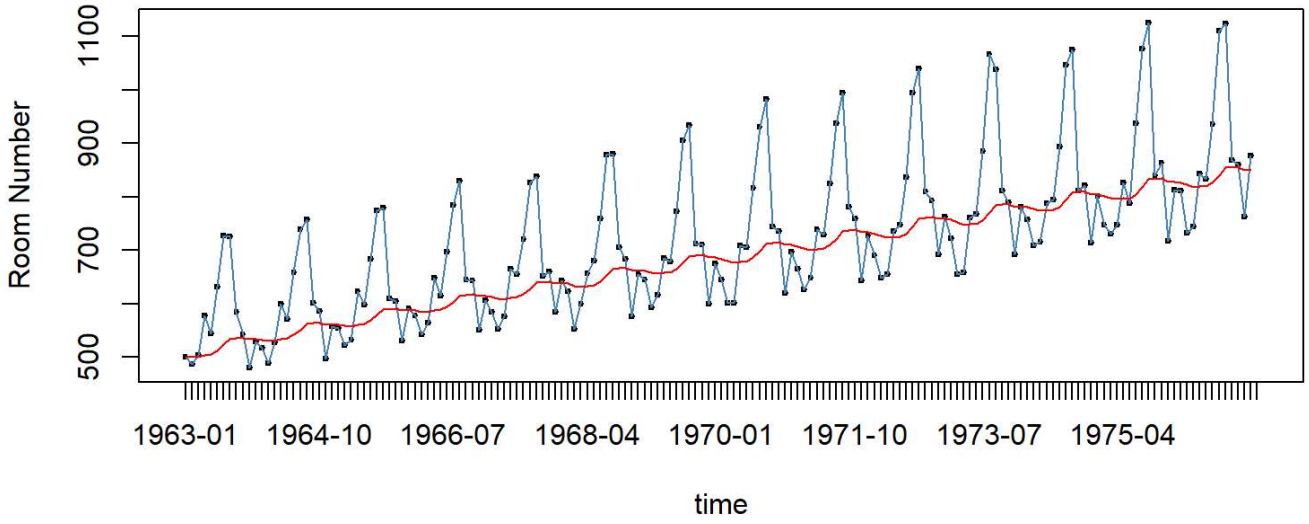
4.2 ES results

- alpha=0.2

```
alpha=0.2 # size of the window of MA
ts_e=ES(ts, alpha)
plot(ts,xlab="time",ylab="Room Number",pch=c(16))
# plot the lines for the original data
lines(ts,type="l",col = "steelblue",xlab="time",ylab="Room Number",pch=c(16))
# plot the lines for the data after moving average
lines(ts$Month,ts_e,type="l",col = "red",xlab="time",ylab="Room Number",pch=c(16))
```



- alpha=0.05



5 Differencing

The difference operator in the discrete field is the analogous of the derivative in the real field. By applying this we can see if we are facing a polynomial of any degree: we just need to apply the function n times (where n is the polynomial degree) to the time series and verify if we get a constant value at the end.

$S_t = y_t - y_{t-n}$, where n is the lag when we do differencing, in our case, we consider it as 1.

```
# Definition of difference operator
difference <- function(ts){
  size = nrow(ts)
  result=vector(mode="numeric",length=size)
  result[1] = ts[1,2]
  for (i in 2:(size)) {
    result[i]=ts[i,2]-ts[i-1,2]
  }
  return(result)
}

ts2 <- difference(ts)

plot(ts$Month,ts2,ts,xlab="time",ylab="Room Number",pch=c(16))
# plot the lines for the original data
lines(ts$Month,ts2,type="l",col = "orange",xlab="time",ylab="Room Number",pch=c(16))
```