Assignment 3 STA457H1F

Mattea Busato

November 15, 2024

1 Exercise - Monthly traffic fatalities in Ontario

1.1 Seasonally adjust the data

Firstly, we want to seasonally adjust the data. To do so, we tune the parameters s.window and t.window of the function stl to achieve a model that gives white noise residuals. Once we achieve residuals close to white noise we have successfully estimated the seasonal and trend components.

We notice immediately that for every value of t.window, s.window="periodical" gives worse residuals (in terms of white noise) than any other numerical value. Moreover, as t.window increases the residuals show that the estimated model fits less and less the variance.

After trying different values for the two parameters, we notice that with s.window=3 and t.window=21 we find the residuals that are the closest to white noise. But, following Cleveland et al. we know that s.window should be odd and at least 7. We conclude that the best parameters we have found are s.window=7 and t.window=21.

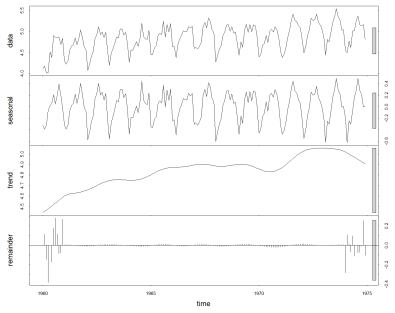

Figure 1: s.window=3 and t.window=21

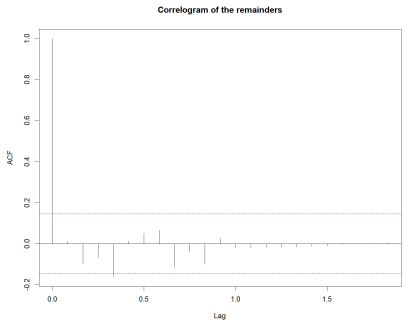

Figure 2: s.window=3 and t.window=21

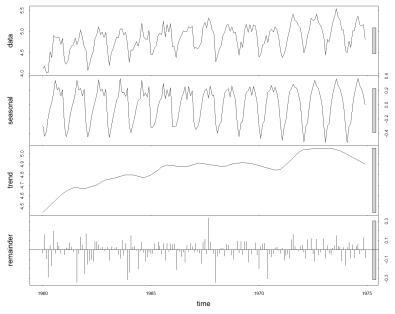

Figure 3: s.window=7 and t.window=21

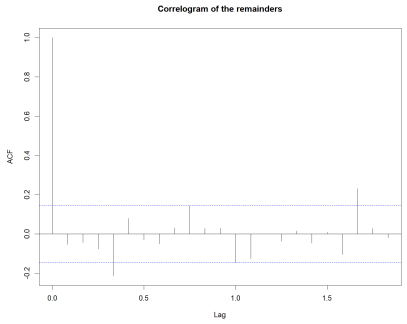

Figure 4: s.window=7 and t.window=21

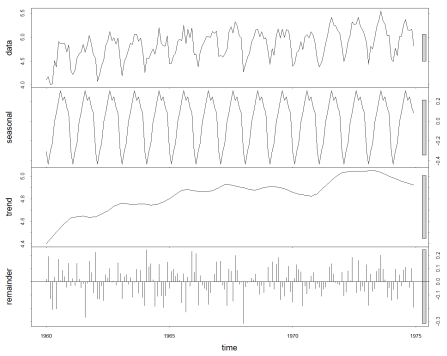

Figure 5: s.window=periodic and t.window=21

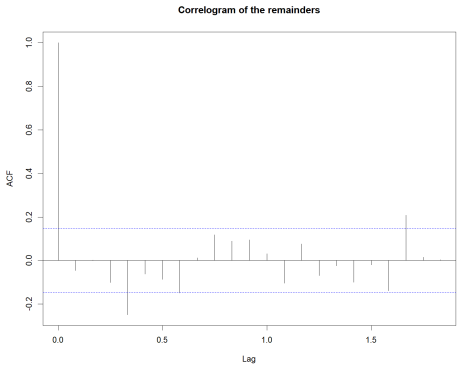

Figure 6: s.window=periodic and t.window=21

1.2 Estimated irregular components

As we have already anticipated in subsection 1), we do expect the residuals to look like white noise when the seasonal component and the trend component are estimated and removed. This is because when we seasonally adjust we are trying to estimate seasonal and trend components that explain the variance of the time series well and that do not leave any substantial variance in the residuals.

Plotting the correlogram of the reminders shows us that they are indeed close to white noise.

1.3 Calendar component

The answer to this question depends on many factors.

If we assume that the number of fatalities is somewhat correlated with the number of holidays and weekend days in a month, as someone might think just by looking at the data in newspapers about the number of fatalities that happen over weekend nights, then we could think that a component that reflects these value per each month would definitely need to be included in the model.

If such holidays and weekend days were a fixed number per month, ie x in January, y in February and so on, then the seasonal component previously estimated would be enough to explain the variance in the time series.

We know though that some holidays do not happen every year on the same date, for example Easter. Also, the number of weekend days per month changes every year. Assuming that on such days there is a higher number of fatalities occurring - that might be caused by elevated alcohol consumption or other factors - then we might have to add this calendar component to the model to better explain the variance.

The stl seasonal adjustment for this type of data, without any additional calendar components, would work only if the effect of such "moving" holidays and amount of weekend days is negligible to the overall fatalities for the month. But this can be decided only by looking at the particular time series you are analyzing.

2 Exercise - Speech recording

2.1 Spectral Density function using an AR model

We try to fit AR models of different orders to the data and see the respective estimated spectral densities.

As we increase order, the spectral density looks more and more rough with added peaks at certain frequencies.

From the AR model chosen with AIC - AR(18) - we see that the most dominant frequencies are around 700, 1400 and 2100.

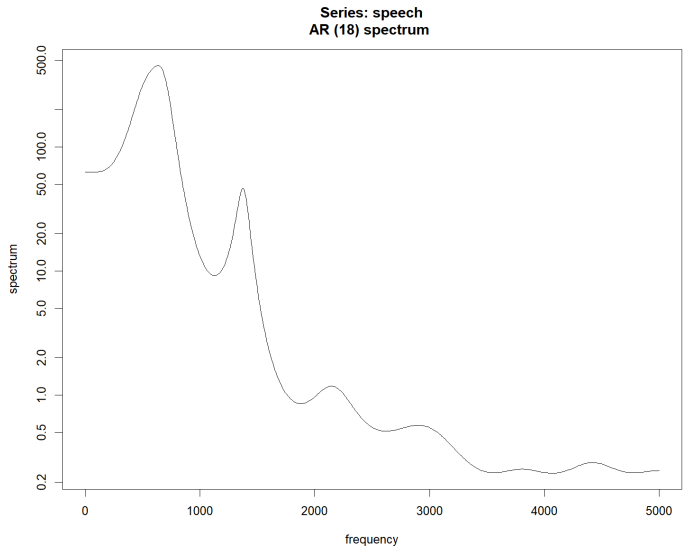

Figure 7: AR(18)

2.2 Spectral Density function using Multitaper

With the Multitaper method, as we increase nw and k, we reduce the variance (due to averaging), but we lose resolution in the frequencies.

We try with different values of nW and k, we notice that at nw=10 and k=18 we can clearly identify the prominent peaks of frequencies at around 800 and around 1500, similar to the ones we have found in part 1).

Surely the estimated spectral densities found in part 1) look smoother, but the dominant frequencies that we identify with the two methods are more or less the same.

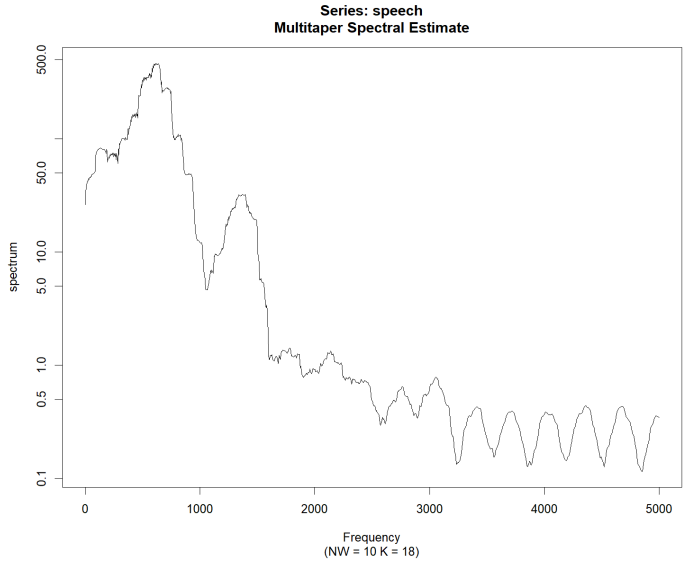

Figure 8: nw=10 and k=18

2.3 Spectral Density function using Parzen's lag window

We try Parzen's estimates with M = 30, 40, 60, 80, 100 to check the peak frequencies at different ratios of smoothness and resolution.

We noticed that the dominant frequencies are shown clearly from M = 60 up. These are the same ones we found earlier, around 700/800 and around 1500.

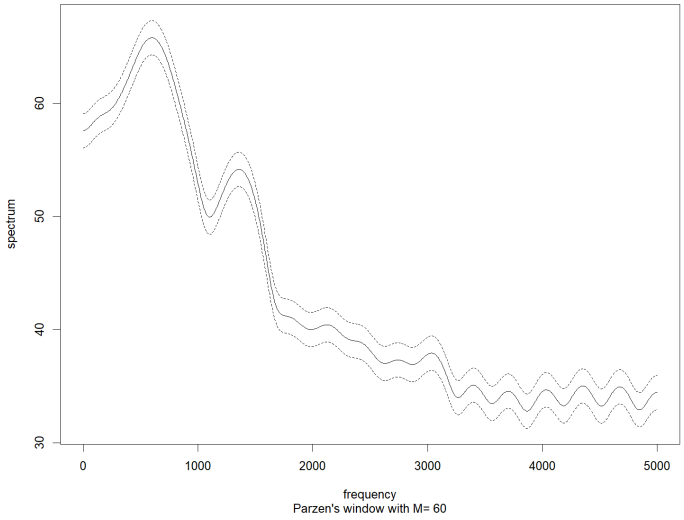

Figure 9: M = 60

2.4 Most dominant frequencies

As we have talked about before, the most dominant frequencies are around 700/800 and 1400/1500. This can actually be expected since each vowel has specific formant frequencies that identify them. For example, the frequencies around 700 Hz and 1400 Hz indicate a vowel sound where the mouth is more open and the tongue position is relatively low, characteristics of vowels like /a/ or /æ/ in English.

A RScript for Exercise 1

Figure 10

```r
fatal <- scan(file.choose())
fatal <- ts(log(fatal), start=c(1960,1), end=c(1974, 12), freq=12)

# 1) Seasonally adjust the data
r <- stl(fatal, s.window=3, t.window=21, robust=T)
plot(r)
time <- r $ time.series
acf(time[,'remainder'], main="Correlogram of the remainders")

r <- stl(fatal, s.window=7, t.window=21, robust=T)
plot(r)
time <- r $ time.series
acf(time[,'remainder'], main="Correlogram of the remainders")

r <- stl(fatal, s.window="periodic", t.window=21, robust=T)
plot(r)
time <- r $ time.series
acf(time[,'remainder'], main="Correlogram of the remainders")

# 2) Check irregular component of one of the estimated parameters
r <- stl(fatal, s.window=7, t.window=21, robust=T)
plot(r)
time <- r $ time.series
acf(time[,'remainder'], main="Correlogram of the remainders")
```

Figure 10:

B RScript for Exercise 2

Figure 11

```
library(multitaper)
speech <- ts(scan(file.choose()),frequency=10000)

# 1) Spectral Density function using an AR model
r <- spec.ar(speech,order=5,method="burg")
r <- spec.ar(speech,order=10,method="burg")
r <- spec.ar(speech,order=20,method="burg")
r <- spec.ar(speech,order=30,method="burg")
r <- spec.ar(speech,method="burg")

# 2) Spectral Density function using Multitaper
r <- spec.mtm(speech, nw=4, k=7)
r <- spec.mtm(speech, nw=10, k=18)
r <- spec.mtm(speech, nw=13, k=24)
r <- spec.mtm(speech, nw=15, k=27)
r <- spec.mtm(speech, nw=20, k=36)

# 3) Spectral Density function using Parzen's lag window
source(file.choose())
r <- spec.parzen(speech,lag.max=30,plot=T)
r <- spec.parzen(speech,lag.max=40,plot=T)
r <- spec.parzen(speech,lag.max=60,plot=T)
r <- spec.parzen(speech,lag.max=80,plot=T)
r <- spec.parzen(speech,lag.max=100,plot=T)
```

Figure 11: