MATH 422 - Final Project Paper

Uyen Le

December 2022

Introduction

Spectral analysis is a fundamental technique in time series that allows us to discover the underlying periodicities. In spectral analysis, we want to know which frequencies matter. A starting tool is to produce a periodogram based on the given sample. The periodogram gives us a rough idea of the spectral components associated with each frequency. Large values of the periodogram correspond to the frequencies that are predominant in the time series. However, the periodogram does not produce a reliable estimate of the population spectral density since it is only computed at the sample level. Nevertheless, we can still estimate the spectral density with a confidence interval produced from smoothing the periodogram and assuming that we have a Chi-square distribution and 2 degrees of freedom.

Typically, in R, we use `mvspec` to produce such a confidence interval for the spectral density. Not only does the `mvspec` function require log = "y" before constructing the confidence interval but the output is also not visually appealing. The output is simply a vertical line indicating how large the confidence interval is and a horizontal line indicating where the baseline is. Moreover, sometimes, even with averaging, the resulting confidence interval is still too wide to be of much use (as shown in Figure 2). Hence, it is suggested that we use a bootstrap approach to reconstruct the true spectral density based on resampling the periodogram of the original time series.

Background

[1] proposed the following algorithm to bootstrap the spectral density of a time series:

1. Centering: Subtract the mean from the data.
2. Initial Estimate: Compute the periodogram $I(\omega)$ based on the original time series. Then with a global bandwidth $h_i > 0$ independent of ω,

calculate a smoothed spectral density

$$\hat{f}(\omega_k, h_i) = \frac{1}{nh_i} \sum_{k=-n}^{n} K\Big(\frac{\omega - \omega_k}{h_i}\Big) I(\omega).$$

where $K(\cdot)$ is a given symmetric, nonnegative function on the real values.

3. Compute and Rescale Residuals: The residuals are computed as

$$\epsilon_k = \frac{I(\omega_k}{\hat{f}(\omega_k, h)}$$

as a result of interpreting the spectral density estimation as a multiplicative regression. The residuals are approximately independent and identically distributed for a large sample size. Next, rescale the residuals

$$\tilde{e}_k = \frac{\hat{\epsilon_k}}{\epsilon}, \text{ for } k = 1, \ldots, n, \text{ and } \epsilon = \frac{1}{n} \sum_{k=1}^{n} \hat{\epsilon}_k$$

to have the residuals center around 1.

4. Resample Residuals: Draw an independent sample of $\tilde{e}_1^*, \ldots, \tilde{e}_n^*$ from $\tilde{e}_1, \ldots \tilde{e}_n$ with replacement.

5. Resample Estimate: Choose a resampling bandwidth g to recalculate a smoothed spectral density $\hat{f}(\omega_k, g)$. Next, compute a new periodogram as

$$I^*(\omega_k) = I^*(-\omega_k) = \hat{f}(w_k, g)\tilde{\epsilon_k}^*.$$

Then with a new bandwidth h, the bootstrap kernel spectral density is calculated as

$$\hat{f}(\omega, h, g) = \frac{1}{nh} \sum_{k=-n}^{n} K\Big(\frac{\omega - \omega_k}{h}\Big) I^*(\omega_k).$$

The estimated spectrum resulting from a large number of bootstrap iterations forms a distribution of the true spectral density that we will use later to construct a confidence interval.

A few notes about the algorithm: In the first step, the data is centered by either detrending or demeaning to get rid of any additional bias in the resampling process. Nonstationary trends should be eliminated in advance since they can introduce extremely low frequency components in the periodogram that tend to obscure the spectrum at higher frequencies. Hence, although centering is optional, it is recommended before computing the periodogram. The rescaling of residuals in step 3 serves the same purpose as recentering the data. We want to keep the bias as low as possible.

In steps 2 and 5, a kernel is simply a weighted average smoother. A few examples of kernels are a Barlett smoother, a Daniell smoother, or a modified

Daniell weighted average (Figures 3 and 4). The paper uses a Barlett kernel with a bandwidth chosen based on experimental results. It is noted that the three bandwidths h_i, g, h do not have to be one or different. They can be anything, but the paper chooses $h_i = g = h$ for the Barlett kernel in their implementation of the algorithm. The authors also mentioned that different kernel choices could result in different results. Guidelines on how to choose the best kernel and bandwidth exist in the literature, but this is beyond the scope of the project.

Finally, the paper uses a traditional distribution estimation technique to compute the upper bound and lower bound of the confidence interval. I'm not sure how they did it, so in this work, we will simply use the percentiles of the bootstrap distribution to build the confidence interval.

Methods

I followed the framework proposed in [1] and [4] and created the function `spec.boots` inside the package of the same name. The purpose of the function is to produce a bootstrap confidence interval of the true spectral density by resampling the periodogram of the original time series.

The function must be given a time series `x`, a number of bootstrap samples `nboot`, and either a kernel or specified spans for the Daniell kernel as input. The function outputs a plot of the resulting confidence interval and returns a dataframe with the lower bound and upper bound associated with the frequencies.

First, the function checks if the arguments are valid, i.e `x` is a time series; `spans` and `kernel` are valid, etc. Next, the function finds the frequencies j/n, where n is the sample size and $j = 0, \ldots, n-1$. Then `spec.boots` centers the data by default using built-in `detrend` function or simply subtracting the mean from the data `x = x - mean(x)`.

Consequently, we compute the periodogram, truncate the second half since it is the same as the first half, then calculate an estimated spectral density given `kernel`.

```
# step 2: initial estimate
I = Mod(fft(x)/sqrt(n))^2
I = I[1:floor(n/2)]
f.hat = kernapply(I, kernel, circular=TRUE)
```

Step 3 compute and rescale the residuals that are simply the periodogram divided by the smoothed spectral density.

```
# step 3: compute and rescale residuals
eps = I / f.hat
scaled_eps = eps / mean(eps)
```

The next step resamples the residuals as well as the bootstrap kernel spectral density an `nboot` number of times and stores the results in a 2D array `f.star.dist`.

```
# step 4-6: resample residuals and bootstrap spectral denisty
f.star.dist = c(c())
for (i in 1:nboot)
{
    scaled_eps.star = sample(scaled_eps, replace=TRUE)
    I.star = f.hat * scaled_eps.star
    f.star = kernapply(I.star, kernel, circular=TRUE)
    f.star.dist[i] = list(f.star)
}
```

Finally, looping through `f.star.dist` and extracting the 2.5 percentile and 97.5 percentile spectrum at every frequency gives us a confidence interval for the true spectral density. The output is a plot of the upper bound and lower bound as well as a baseline indicating the median of the spectral density.

An Example

An example of using `spec.boots` on the `AirPassengers` dataset is demonstrated in the `demo.Rmd` file and below. It is required to install the package from Github.

```
install.packages("devtools")
devtools::install_github("uyenle-gh/spec.boots")
data("AirPassengers")
tsplot(AirPassengers, col=4, lwd=2)
spec.boots(AirPassengers, 1000, c(5,5))
```

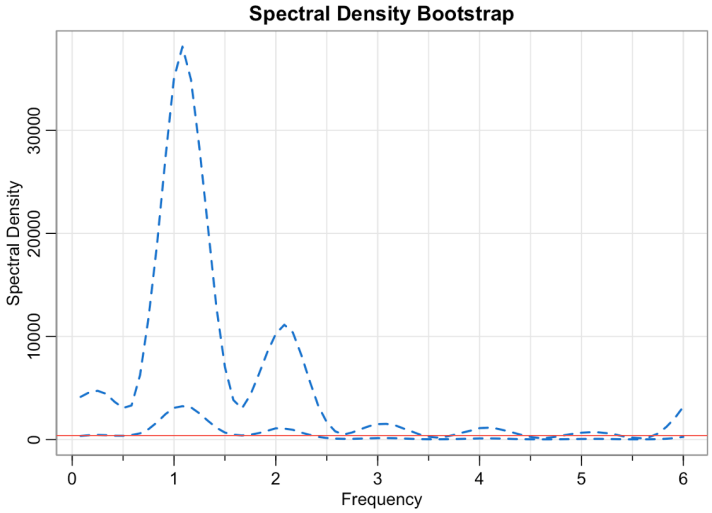

Figure 1: Using `spec.boots` on `AirPassengers`.

The `spans = c(5,5)` is specified, so the function will use a modified Daniell kernel with the given bandwidth throughout the resampling process. The result should be the same if we give the function a modified Daniell kernel as follows:

```
kd = kernel("daniell", c(5,5))
spec.boots(AirPassengers, 1000, kd) # same result
```

Engineering Issues

Although this is a computational project, the hardest part lies in researching and covering the literature. I was overwhelmed at first because the papers talked a lot about the math that we didn't focus on in class. After a few reads, I found the algorithm straightforward but wasn't entirely sure that our way of computing the periodogram and the smoothed spectral density matched theirs. I tried a few different methods, and this is the best of all.

Additionally, I had never programmed in R, and to be honest, I didn't like R when I first started using it in DA101 two years ago. R is very different from other common programming languages like Python and C++. Another thing is that there is no available code specific to bootstrapping spectral density that I can refer to, so I have to base on how they built `mvspec` and `tsboots` to build my own function. Lastly, creating the package was much more time-consuming than I expected, so I didn't have much time to create a test file to cover all the cases as well as implement a different type of bootstrapping. Nevertheless, I am really happy with the result and become to like R much better now as I know more about it.

Conclusion

The `spec.boots` function successfully produces a confidence interval for the population spectral density based on resampling the periodogram of the original time series. The kernel choice has a big impact on the final result, so future directions include bootstrapping in the time domain (like the tapered block bootstrap or the autoregressive-aided periodogram bootstrap in [4]). In the future, I will continue refining my work and bringing in more methods from other studies.

References

[1] Franke, J., and W. Hardle. *On Bootstrapping Kernel Spectral Estimates.* The Annals of Statistics, vol. 20, no. 1, 1992, pp. 121–45. JSTOR. http://www.jstor.org/stable/2242153.

[2] *tsbootstrap: Generate time-series bootstrap replicates.* resamplr: Resampling Methods. R Documentation. https://rdrr.io/github/jrnold/resamplr/man/tsbootstrap.html.

[3] Stoffer, David and Poison, Nicky. m*vspec: Univariate and Multivariate Spectral Estimation.*. astsa: Applied Statistical Time Series Analysis. R Documentation. 2017. https://rdrr.io/cran/astsa/man/mvspec.html.

[4] Zoubir, Abdelhak M.. *Bootstrapping spectra: Methods, comparisons and application to knock data.* Signal Process. 90, 5. May, 2010, 1424–1435. https://doi.org/10.1016/j.sigpro.2009.11.030.

Appendix

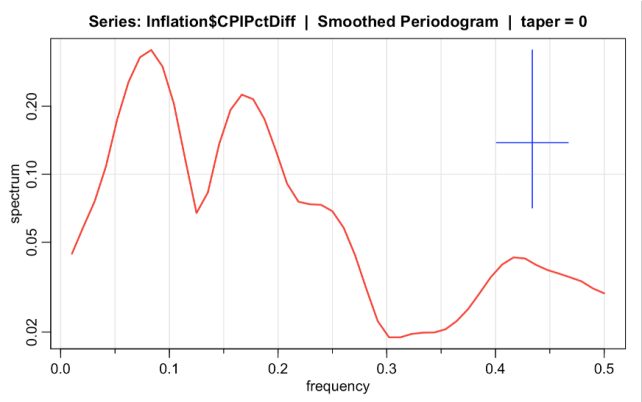

Figure 2: The `mvspec` command returns a too wide confidence interval for the spectral density of the CPI returns to be interpretable.

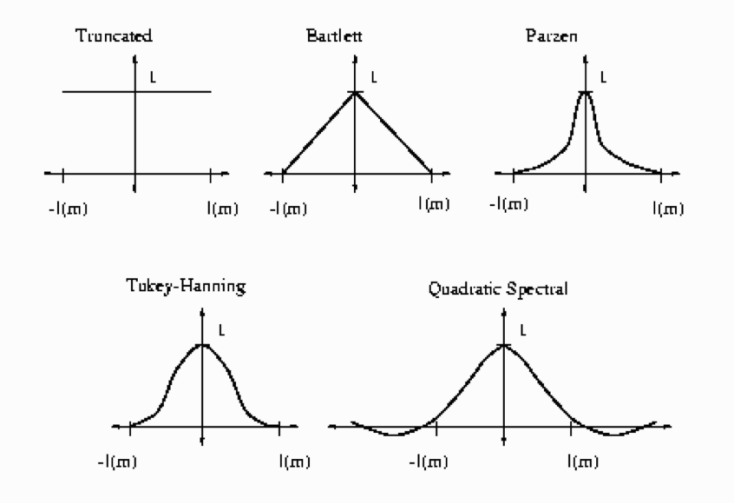

Figure 3: Examples of kernels.

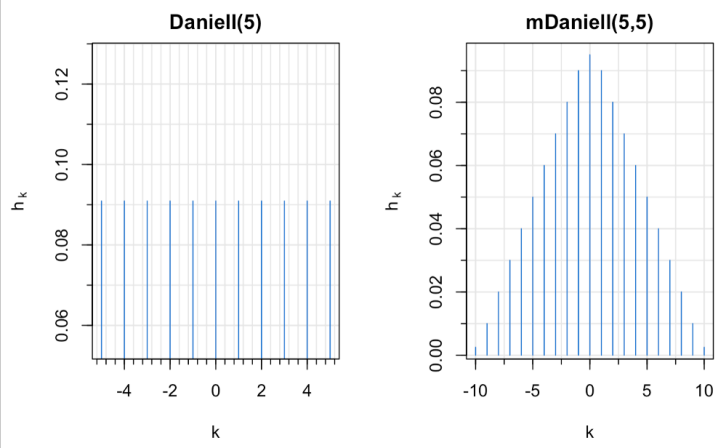

Figure 4: Normal Daniell kernel and modified Daniell kernel.

Usage

```
spec.boots(
  x,
  nboot,
  spans = NULL,
  kernel = NULL,
  detrend = TRUE,
  demean = FALSE,
  base.stat = median,
  alpha = 0.95
)
```

Arguments

`x`	a time series
`nboot`	the number of bootstrap series to compute
`spans`	specify smoothing
`kernel`	specify kernel
`detrend`	if TRUE, data is detrended first (TRUE as default)
`demean`	if TRUE, data is demeaned first (FALSE as default)
`base.stat`	a function applied to return the baseline statistic of interest (median as default)
`alpha`	a confidence interval (0.95 as default)

Figure 5: Details on how to use `spec.boots`.