On Bootstrapping Spectral Estimates

Beniamino Hadj-Amar

April 23, 2016

Setting the frequency

Regardind setting the frequency, without messing up. This is true when n is even.

```
# Regarding the frequencies:
n <- length(data)
n.tilda <- n/2
k <- 0:(n.tilda - 1)
freq.rad <- 2*pi*k/n
freq <- k/n

# Finding the period:
# Let's say that I has a pick at position 13, the relative period is given by
1/freq[13]
2*pi/(freq.rad[13])

# Also remember to set I(0) = 0
```

1 Periodogram and Smoothed Periodogram

Consider a discrete-time, real-valued stationary time series $\{X_t\}_{t\geq 1}$. Let's consider realisations $x_1, \ldots, x_n$ of the process $\{X_t\}_{t\geq 1}$. For simplicity we assume that n is even, and define $\widetilde{n} = n/2$. The periodogram is the Fourier transform of the empirical autocovariance function and can be written as:

$$I(\omega_k) = \frac{1}{2\pi} |\sum_{t=1}^{n} x_t \exp(-i\omega_k t)|^2,$$

where $\omega_k = 2\pi \mathrm{k}/n$, $\mathrm{k} = 1, \ldots, \widetilde{n}$.

The code to compute the periodogram, given a time series is given below:

```
# Function: Periodogram
periodogram <- function(data) {
  n <- length(data)
  I <- (abs(fft(data))^2)/(n*2*pi)
  return(I)
}
```

The periodogram is a unbiased but inconsistent estimator of the frequency spectrum. A consistent estimator is obtained through smoothing techniques by a kernel spectral estimate of the form

$$\boxed{\hat{f}_b(\omega_k) = \frac{\sum_{j=-\tilde{n}}^{n-1} K_b(\omega_j - \omega_k) I(\omega_j)}{\sum_{l=-\tilde{n}}^{n-1} K_b(\omega_l - \omega_k)}} \tag{1}$$

where $K_b(x)$ is a (scaled) Gaussian kernel:

$$K_b(x) = \frac{1}{b}\frac{1}{\sqrt{2\pi}}\exp\left\{\frac{(x/b)^2}{2}\right\} \tag{2}$$

```r
# Function: Gaussian Kernel
gaussian.kernel <- function(x, b = 1, scaled = FALSE) {

  if(scaled == TRUE) {
    out <- (1/b) * (1/sqrt(2*pi)) * exp(-((x/b)^2)/2)
    return(out)
  }

  out <- (1/sqrt(2*pi)) * exp(-(x^2)/2)
  return(out)
}
```

Furthermore, $I(0) = 0$, and $I(\omega_{-k}) = I(\omega_k)$. The bandwith b is a positive paramater that controls the amount of smoothing imposed over $I(\omega)$.

The code to compute the smoothed periodogram in 1 is given below:

```r
# Function: Smoothed periodogram for single frequency
single.smoothed.periodogram <- function(x, I, b) {

  # Auxiliary useful variables
  n <- length(I)
  n.tilda <- n/2
  k <- seq(from = -n.tilda, to = n - 1)
  I.aux <- c(I[n.tilda:1], 0,  I[1:(n - 1)])
  freq.aux <- 2*pi*k/n

  # Kernel coefficients
  temp <- gaussian.kernel(freq.aux - x, b, scaled = TRUE)

  # Evaluating smoothed periodogram
  numerator <- temp %*% I.aux
  denominator <- sum(temp)
  I.smoothed <- numerator/denominator

  return(I.smoothed)
}

# Function: Smoothed periodogram
smoothed.periodogram <- function(freq, I, b) {
  out <- c()
  for(i in 1:length(freq)) {
    out[i] <- single.smoothed.periodogram(freq[i], I, b)
  }
  return(out)
}
```

1.1 Selecting the bandwith b

The optimal value of the bandwith b is $c \times n^{-1/5}$, for some positive constant c. It has been proposed to choose b to minimise a risk function, which is an unbiased estimator of the residual sum of square.

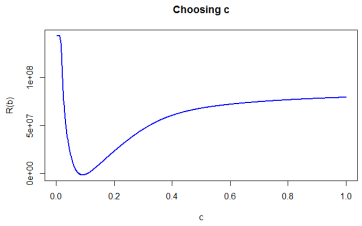
$$\hat{R}(b) = \sum_{k=0}^{\tilde{n}-1} \left\{ I(\omega_k) - \hat{f}_b(\omega_k) \right\}^2 - \frac{1 - 2W_b}{2} \sum_{k=0}^{\tilde{n}-1} I(\omega_k)^2, \tag{3}$$

where $W_b = K_b(0) / \sum_{j=-\tilde{n}}^{n-1} K_b(\omega_j)$.

The code to compute 3 is given by

```
### Function: unbiased estimate for Residual Sum of Squares (RSS)
unbiased.RSS <- function(b, I) {

  # n, \tilde{n}
  n <- length(I)
  n.tilda <- n/2
  # Frequencies
  k <- 1:n.tilda
  freq.rad <- 2*pi*k/n

  # Smoothed.periodogram
  I.smooth <- smoothed.periodogram(freq.rad, I, b = b)

  # Auxiliary variables
  k.aux <- seq(from = -n.tilda, to = n - 1)
  freq.aux <- 2*pi*k.aux/n

  # Constant for bias
  K.0 <- gaussian.kernel(0, b = b, scaled = T)
  norm.const <- sum(gaussian.kernel(freq.aux, b = b, scaled = T))
  W.b <- K.0/norm.const

  # RSS
  RSS <- sum((I[1:n.tilda] - I.smooth)^2)
  # Bias
  bias <- ((1 - 2*W.b)/2) * sum(I[1:n.tilda]^2)
  # Unbiased estimate
  out <- RSS - bias

  return(out)
}
```

In pratice to find b, we compute $\hat{R}(b)$ for a grid of value of c (let's say between 0 and 1), and we select the value of c which minimises this risk function. An example of this is given in the figure below, by using the dataset `sunspotz`.



To make the life easier we make a function to find the value of c, and we test the performance of the spectral smoothing, and the elictation of the bandwith. The code and the plots are given below:

```r
source("Kernel_Smoothing_Spectrum.R")

library(astsa)
data("sunspotz")

# Testing on Sunspotz Data (considering n even)
x <- as.vector(sunspotz);
x <- x[-1]
n <- length(sunspotz)

# Frequency (radiant)
n.tilda <- n/2
k <- 0:(n.tilda - 1)
freq.rad <- 2*pi*k/n

# Periodogram, and setting I(0) = 0
I <- periodogram(x)
I[1] <- 0

# Smoothed periodogram
# Notice: to evaluate smoothed.periodogram we need to use freq.rad
I.smooth <- smoothed.periodogram(freq.rad, I, b = 0.14)

# Plotting periodgram and smoothed periodogram
plot(freq.rad, I[1:(n.tilda)], type = "l")
lines(freq.rad, I.smooth[1:(n.tilda)], type = "l",
      col = "red", lwd = 2)

# We want to find the c, for which the unbised RSS(b) is minimised.
c <- find.c(I)
c;

# Therefore the optimal b is given by:
b <- c * (n ^ (-1/5)); b
I.smooth <- smoothed.periodogram(freq.rad, I, b = b)

# The plotted results:
plot(freq.rad, I[1:(n.tilda)], type = "l", ylab = "Power spectrum")
lines(freq.rad, I.smooth[1:(n.tilda)], type = "l",
      col = "red", lwd = 2)
```

where

```r
# Function: find c that minimise the unbised RSS
find.c <- function(I) {

  # Length time series
  n <- length(I)
  # Grid
  c <- (1:1e3)/1e3
  b <- c * (n^(-1/5))

  # Risk values
  risk.values <- c()
  for(i in 1:length(c)) {
    risk.values[i] <- unbiased.RSS(b = b[i], I)
  }

  # Values of C that minimises the risk value
  out <- c[which(risk.values == min(risk.values))]

  return(out)
}
```

The spectrum and the smoothed spectrum are given in Figure 1

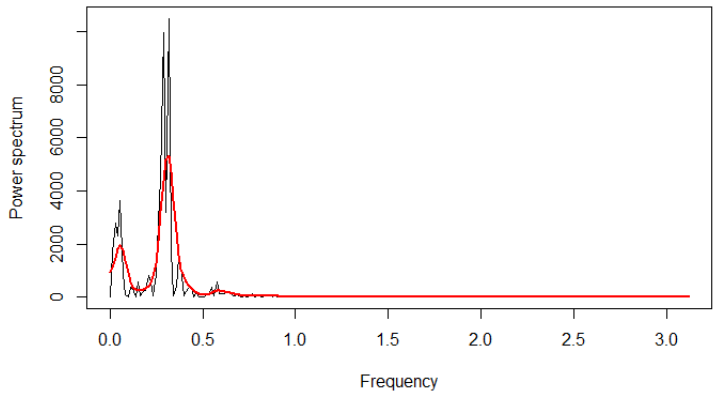


Figure 1: Black line: periodogram; Red Line: smoothed spectrum, with $c = 0.091$, and therefore $b = 0.0267$

2 Bootstrap Spectral Estimates

Recall the important result $\frac{2I(\omega_k)}{f(\omega_k)} \sim \chi^2(2)$. It follows immediately that

$$I(\omega_k) = f(\omega_k)\epsilon_k, \quad \text{where } \epsilon_k \sim \text{Expo}(1). \tag{4}$$

We use this relationship to interprete the spectral estimation problem as an approximate multiplicative regression problem.

We consider the following procedure for getting a bootstrap approximation of $\hat{f}_b(\omega_k)$. We define this method as *Spectral Resampling* (SR).

- **Step 1**

  We choose an initial bandiwth $b_\dagger = c \cdot n^{-1/4}$. We estimate the residuals $\hat{\epsilon}_k$, $k = 1, \ldots, \tilde{n}$ of 4 as

  $$\hat{\epsilon}_k = \frac{I(\omega_k)}{\hat{f}_{b_\dagger}(\omega_k)}, \quad k = 1, \ldots, \tilde{n}$$

  We rescale the empirical residuals by considering

  $$\tilde{\epsilon}_k = \frac{\hat{\epsilon}_k}{\hat{\epsilon}}, \quad k = 1, \ldots, \tilde{n}, \quad \text{where } \hat{\epsilon} = \frac{1}{\tilde{n}} \sum_{j=1}^{\tilde{n}} \tilde{\epsilon}_j$$

- **Step 2**

  We draw independent bootstrap residuals $\epsilon_1^*, \ldots, \epsilon_{\tilde{n}}^*$ from the empirical distribution of $\tilde{\epsilon}_1, \ldots, \tilde{\epsilon}_{\tilde{n}}$

  Following 4 we define bootstrap periodogram values as

  $$I^*(\omega_k) = I^*(\omega_{-k}) = \hat{f}_{b_\ddagger}(\omega_k)\epsilon_k^*, \quad k = 1, \ldots, \tilde{n}$$

  ,

where $b_\ddagger = c \cdot n^{-1/6}$.

Finally we obtain a bootstrap spectral estimate as

$$\hat{f}_b^*(\omega) = \frac{\sum_{k=-\tilde{n}}^{n-1} \Big\{ K_b(\omega_k - \omega) I^*(\omega_k) \Big\}}{\sum_{l=-\tilde{n}}^{n-1} \Big\{ K_b(\omega_l - \omega) \Big\}}$$

where $b = c \cdot n^{-1/5}$. Notice that in order to evaluate $\hat{f}_{b_\ddagger}(\omega_k)$ for $k > \tilde{n}$, we use the fact that

$$\hat{f}_{b_\ddagger}(\omega_{[\tilde{n}+k]}) = \hat{f}_{b_\ddagger}(\omega_{[\tilde{n}-k]}), \quad |k| > \tilde{n}.$$

We now give the code to implement the SR method. We need a couple of functions to evaluate the final bootstrap spectral estimate:

```
### Function: single estimate for bootstrap spectrum
single.f.star <- function(x, I.smooth, residuals, b) {

  # Setting length parameters
  n <- length(I.smooth) * 2
  n.tilda <- length(I.smooth)

  # Parameters
  k <- seq(from = -n.tilda, to = n - 1)
  freq.aux <- 2*pi*k/n

  # Auxiliary objects
  residuals.aux <- c(residuals[n.tilda:1], 0, residuals[1:n.tilda],
                     residuals[(n.tilda - 1) : 1])
  I.smooth.aux <- c(I.smooth[n.tilda:1], 0, I.smooth[1:n.tilda],
                    I.smooth[(n.tilda - 1):1])

  # Bootstrap periodogram
  I.star <- I.smooth.aux * residuals.aux

  # Kernel coefficients
  temp <- gaussian.kernel(freq.aux - x, b, scaled = TRUE)

  # Evaluating bootstrap spectral estimates
  numerator <- temp %*% I.star
  denominator <- sum(temp)
  I.smoothed.BR <- numerator/denominator

  return(I.smoothed.BR)
}

### Function: vector estimate for bootstrap periodogram
f.star <- function(freq, I.smooth, residuals, b) {
  out <- c()
  for(i in 1:length(freq)) {
    out[i] <- single.f.star(freq[i], I.smooth, residuals, b)
  }
  return(out)
}
```

Finally the full SR algorithm is given below:

```
### Function: Spectral Resampling algorithm
Spectral.Resampling <- function(data, c, R) {

  # Setting length parameters
  n <- length(data)
  n.tilda <- n/2
  k <- 0:(n.tilda - 1)
  freq.rad <- 2*pi*k/n

  # Setting bandwiths
  b.start <- c * (n ^ (-1/4))
  b.intermediate <- c * (n ^ (-1/6))
  b.final <- c * (n ^ (-1/5))

  # Output boostrap spectral estimate
  I.smooth.BS <- matrix(NA, nrow = n.tilda, ncol = R)

  for(r in 1:R) {

    # STEP 1:
    I <- periodogram(data)
    I[1] <- 0
    I.smooth <- smoothed.periodogram(freq.rad, I, b = b.start)
    residuals <- I[1:n.tilda]/I.smooth[1:n.tilda]
    residuals.norm <- residuals/(sum(residuals)/n.tilda)


    # STEP 2:
    residuals.BS <- sample(residuals.norm, size = n.tilda,
                           replace = TRUE, prob = rep(1/n.tilda, n.tilda))

    I.smooth.intermediate <- smoothed.periodogram(freq.rad, I, b = b.intermediate)

    # Final boostrap estimate:
    I.smooth.BS[, r] <- f.star(freq.rad, I.smooth.intermediate,
                               residuals = residuals.BS, b = b.final)
  }
  class(I.smooth.BS) <- "SR"
  return(I.smooth.BS)
}
```

In order to see the performance of the algorithm, we create a function to obtain confidence intervals, and plot average spectrum and the confidence bands.

```
### Function: Get Average, and Confindence Intervals of SR
###           and an optional plot.

Spectral.Resampling.CI <- function(SR.sample) {

  # Getting dimensions and setting frequencies
  n.tilda <- nrow(SR.sample)
  n <- n.tilda*2
  k <- 0:(n.tilda - 1)
  freq.rad <- (2*pi*k)/ n

  # Checking input
  if(class(SR.sample) != "SR")
    stop("input not of class SR")

  # Creating confidence intervals for spectrum estimate
  conf.int <- matrix(NA, nrow = n.tilda, ncol = 2)
  for(i in 1:n.tilda) {
    conf.int[i, ] <- quantile(SR.sample[i, ], probs = c(.05, .95))
  }
```

```
  # Average
  mean.spectrum.SR <- rowMeans(SR.sample)

  # Auxiliary data.frame for plotting
  dat <- data.frame(freq = freq.rad, spec = mean.spectrum.SR,
                    ci.up = conf.int[, 2], ci.low = conf.int[, 1])
  # Plotting mean and C.I
  p <- ggplot() +
    geom_line(data = dat,
              aes(x = freq.rad, y = spec, linetype = "c"),
              size = 1.0, colour = "red") +
    geom_line(data = dat,
              aes(x = freq.rad, y = ci.up),
              size = 0.6, colour = "black", linetype = "dashed") +
    geom_line(data = dat,
              aes(x = freq.rad, y = ci.low),
              size = 0.6, colour = "black", linetype = "dashed") +
    xlab("Frequency") + ylab("Power Spectrum") +
    ggtitle("Spectral Resampling (SR)") +
    theme(legend.position = "none",
          plot.title = element_text(size = 19))


  return(list(spec = mean.spectrum.SR,
              ci.up = conf.int[, 2], ci.low = conf.int[, 1],
              plot = p))
}
```

Plot of mean of the bootstrap sample for **sunspotz** dataset and its relative confidence intervals are given in Figure 2

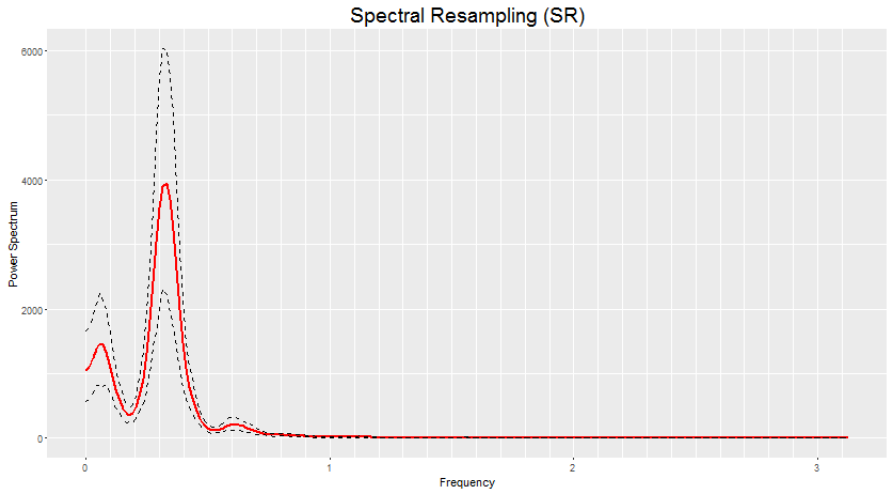


Figure 2: Black line: mean sample SR; Red Lines: 95 % confidence intervals

It is also useful to have an idea about the distribution of the length estimates for the largest peaks in the spectrum, for the SR method. Therefore we give a function to do this job.

```
### Function: distribution of the peaks of SR sample.
#             This function also provides an histogram.
SR.distribution <- function(SR.sample, n.peaks) {

  # Checking input
  if(class(SR.sample) != "SR")
    stop("input not of class SR")
```

```r
  # This library implement the function findpeaks
  library(pracma)

  # Setting dimensions and frequency parameters
  R <- dim(SR.sample)[2]
  n.tilda <- dim(SR.sample)[1]
  n <- n.tilda * 2
  k <- 0:(n.tilda - 1)
  freq.rad <- 2*pi*k/n

  # Largest peaks in each bootstrap sample
  counts <- matrix(NA, nrow = n.peaks, ncol = R)

  # Getting largest peaks for each bootstrap sample
  for(n in 1:n.peaks) {
    for(r in 1:R) {
      counts[n, r] <- freq.rad[findpeaks(SR.sample[, r])[n, 2]]
    }
  }

  # Plotting histogram
  dat <- data.frame(counts = c(counts))
  p <- ggplot(data = dat,
              aes(dat$counts)) +
    geom_histogram(fill = "blue", col = "black", breaks = seq(0, pi, by = .04)) +
    coord_cartesian(xlim = c(0, pi)) +
    scale_x_continuous(minor_breaks = seq(0, pi, 0.1)) +
    theme(plot.margin = unit(c(1,1,1,1), "cm")) +
    xlab("Frequency") + ylab("Power Spectrum")

  return(list(counts = counts, plot = p))
}
```

Figure 3 show the distribution of the bootstrap sample for the three largest peaks in the spectrum.

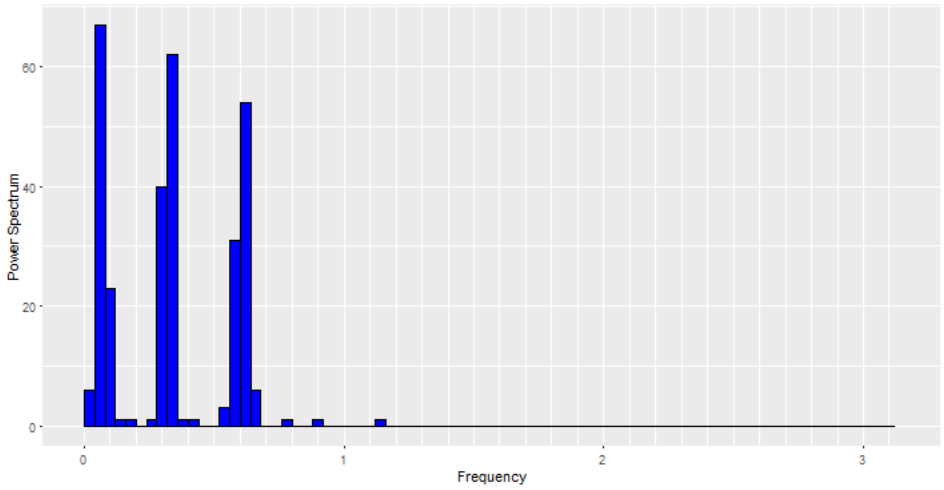


Figure 3: Frequency length estimates for the three largest peaks in spectrum