I presume the reader is familiar at least with the concept of the Fourier transform. Given a function of time $h(t)$, the Fourier transform of h is

$$H(f) = \int_{-\infty}^{\infty} h(t) e^{2\pi i f t}\, dt$$

This is a representation of h in the frequency domain. If t is measured in seconds, then f is measured in cycles per second, or Hertz. There are interesting relations between $H(f)$ and $H(-f)$, but I skip them because they are not essential to the purpose of this entry.

The most important theorem regarding the total power of a signal is Parseval's theorem:

$$\text{total power} \equiv \int_{-\infty}^{\infty} |h(t)|^2\, dt = \int_{-\infty}^{\infty} |H(f)|^2\, df$$

The power spectral density (PSD) of h is defined as

$$P_h(f) := |H(f)|^2 + |H(-f)|^2$$

for $0 \leq f < \infty$. When $h(t)$ is real, the two terms are equal and thus we have

$$P_h(f) = 2|H(f)|^2$$

One sometimes sees $P_h(f)$ without this factor of two, but this is only valid when dealing with two-sided spectral densities. We will always deal with the one-sided spectral density.

Sampled data is discrete, but so far we have discussed continuous functions. We need to provide a few concepts before we can extend the Fourier transform to discrete quantities.

For any sampling interval Δ, there is a special frequency called the Nyquist frequency, defined

$$f_c := \frac{1}{2\Delta}$$

If a sine wave with frequency f_c is sampled with sampling rate Δ, then the distance between a peak and a negative trough value is f_c.

It is quite easy to observe that if we define the sampling rate as $f_s := \frac{1}{\Delta}$, we have $f_c = \frac{1}{2} f_s$. Sometimes this alternative formulation is given.

The sampling theorem states that if a function $h(t)$ sampled with a rate Δ happens to be bandwidth limited to frequencies smaller in magnitude than f_c, then $h(t)$ is completely determined by its samples h_n, and

$$h(t) = \Delta \sum_{n=-\infty}^{\infty} h_n \frac{\sin\left(2\pi f_c (t - n\Delta)\right)}{\pi (t - n\Delta)}$$

This is nice because it shows that the "information content" of a function that is bandwidth-limited can be totally captured through discrete sampling.

The downside of the theorem is that, when the signal being sampled is not bandwidth limited, the spectral density outside the range $-f_c < f < f_c$ is spuriously movied into that range. This is called *aliasing.* Aliasing can be avoided if one imposes a bandwidth limit artificially—e.g. via the sampling method—or one samples at a sufficiently small Δ.

The question we know face is that of estiamting $H(f)$ via a discrete sample $h_1, \ldots, h_N$.

With N input values we may only output N values; thus, instead of estimating $H(f)$ for $f \in [-f_c, f_c]$, we may seek estimates only at

$$f_n := \frac{n}{N\Delta}$$

with $n = -\frac{N}{2}, \ldots, \frac{N}{2}$. The equation above gives us the *frequency domain* of our estimation.

Of course, $H(f)$ is approximated via a discrete sum:

$$H_n := \sum_{k=0}^{N-1} h_k e^{2\pi i k n / N}$$

Observe that H_n, the *Discrete Fourier Transform*, is independent of any parameter other than the sequence $\{h_n\}$, including Δ. When interpreting the result in terms of the sampling frequency, the relationship between the continuous and the discrete Fourier transform is

$$H(f_n) \approx \Delta H_n$$

The discrete form of Parseval's theorem is

$$\sum_{k=0}^{N-1} |h_k|^2 = \frac{1}{N} \sum_{n=0}^{N-1} |H_n|^2$$

There is a famous algorithm, the *Fast Fourier Transform*, for computing the discrete Fourier transform of a vector. It is a very interesting algorithm that is covered in detail in the book I have referenced. I will not explain it here.

1 Computing the power spectrum

The spectral analysis of EEG data typically refers to two things: *(1)* the power spectrum and *(2)* its spatio-temporal distribution. I will not concern myself here with spatio-temporal analysis and will only provide simple R functions that estimate the power spectrum. I will provide this functions first with a simulated signal and then use them in an EEG fragment.

The main difficulty in power spectrum density (PSD) estimation over EEG data is the lack of a standard. Different researchers use different methods, and, as I will attempt to show, different methods produce different outputs.

Let's start from the basis and let's simulate the sampling of a simple signal wave in R. The signal we are to simulate is

$$h(t) = \sin(2\pi \times 100) + 4\sin(2\pi \times 110)$$

This is, a signal that is a mixture of two sine waves with amplitudes 1 and 4, respectively, and frequencies 100 and 110, respectively.

```
set.seed(123)

# Parameters
sampling_rate <- 500 # Sampling rate (samples per second)
duration <- 5-1/sampling_rate          # Duration of the signal in seconds
frequencies <- c(100, 110)  # Frequencies of the sine waves (in Hz)
amplitudes <- c(1, 4)   # Amplitudes of the sine waves

# Generate time vector
t <- seq(0, duration, by = 1/sampling_rate)
```

```
n <- length(t)

# Initialize signal as zeros
signal <- rep(0, n)

# Create a signal as a mixture of sine waves
for (i in seq_along(frequencies)) {
  frequency <- frequencies[i]
  amplitude <- amplitudes[i]

  # Generate the sine wave component
  sine_wave <- amplitude * sin(2 * pi * frequency * t)

  # Add the sine wave component to the signal
  signal <- signal + sine_wave
}
```

The amplitude spectrum of the signal is given by

$$P(f) = \frac{2|H(f)|}{N}$$

which is easily computed in R.

```
my_amplitude <- function(x, sampling_rate){
    N <- length(x)
    ft <- abs(fft(x)) # Absolute value of the FFT.
    ft <- ft[1:(N/2 + 1)] # Make one sided.
    freq <- c(0:(length(ft)-1))*sampling_rate/length(x) # Compute the frequency
    amplitude <- 2/N * ft
    return(list(freq = freq, amplitude = amplitude))
}
```

The PSD is defined as

$$P(f) = \frac{2|H(f)|^2}{W}$$

where $W = \sum_{i=0}^{N-1} w_i^2$ and $w_0, \ldots, w_{n-1}$ is the window function used. If no window is used, this is equivalent to a rectangular window and $W = N$. In R, this is again straightforward to compute:

```
my_psd <- function(x, sampling_rate, han = TRUE){
    N <- length(x)
    if (han){
        hann <- gsignal::hann(N) # Hanning window
        x <- x * hann
        normalization <- 2/(sum(hann^2))
    }else{
        normalization <- 2/(N) # Rectangular window
    }
    ft <- abs(fft(x))
    ft <- ft[1:(N/2 + 1)] # Make one sided
    freq <- c(0:(length(ft)-1))*sampling_rate/length(x) # One sided fft

    power <- ft^2 * normalization
    return(list(spec = power, freq = freq))
}
```

Sometimes, one sees

$$P(f) = \frac{2|H(f)|^2}{W \times f_s}$$

It is important to note that f_s here is a scaling factor that translates from units of power to units of power/Hz. This is useful when comparing powers from signals with different sampling rates. The power spectrum (in decibels) of our signal, with and without a Hanning window, is the following.

Lastly, I provide a quick implementation of Welch's method. Welch's method consist of splitting the signal $h(t)$ into K overlapping segments of length L, where the overlap is defined as a percentage of the segment length (and hence falls within 0 and 1). A slightly modified estimation of the power spectral density is computed for each segment, and then the estimations of all segments are averaged. The estimation in each segment, according to the original paper (https://ieeexplore.ieee.org/document/1161901), is done as follows:

- For each segment $s_i(t)$ of length L, compute $F_i := \frac{1}{L}S_i(f)$, where $S(f)$ is the Fourier transform of s_i, and $f = \frac{n}{L}$ for $n = 0, 1, \ldots, \frac{L}{2}$.
- Compute $U := \frac{1}{L}\sum_{i=0}^{L-1} w_i^2$ with $w_0, \ldots, w_{L-1}$ the window function.
- Compute $\frac{L}{U}|F_i|^2$

This results in the power spectrum of the segment in question. If everything is put together, and we add a factor of two to account for the one-sided spectrum (this is not the case for the original paper), the power spectrum of each segment is

$$A_i = \frac{2L}{U}\left[|\frac{2}{L}S_i(f)|\right]^2 = \frac{2}{LU}|S_i(f)|^2 = \frac{2|S_i(f)|^2}{\sum w_i^2}$$

This expression is prettier to program. In R,

```
overlaps <- function(signal, L, D){
    # Calculate the number of segments
    D <- L * D
    M <- ceiling((length(signal) - L) / (L - D)) + 1

    # Initialize a list to store segments
    segments <- list()

    # Create overlapping segments
    for (i in 1:M) {
      start <- (i - 1) * (L - D) + 1
      end <- start + L - 1
      if (end > length(signal)) {
        break
      }
      segments[[i]] <- signal[start:end]
    }
    return(segments)
}

my_pwelch <- function(signal, sampling_rate, L, D){


    modified_periodogram <- function(segment){
        L <- length(segment) # L to distinguish that this is the length of a seg
        hann <- gsignal::hann(L) # Hanning window
        x <- segment * hann
        ft <- fft(x) # A(n)
        ft <- ft[1:(L/2 + 1)] # Lake one sided
        w <- 2*abs(ft)^2/sum(hann^2)
        freq <- c(0:(length(ft)-1))*sampling_rate/L
        return( list(freq = freq, spec = w) )
    }

    segments <- overlaps(signal, L, D)
    m_periodograms <- lapply(segments, modified_periodogram)
    spectrums <- lapply(m_periodograms, function(x) x$spec)
    n <- length(m_periodograms) # Number of segments
    avg_spectrum <- 1/n * rowSums(do.call(cbind, spectrums))
    return ( list(freq = m_periodograms[[1]]$freq, spec = avg_spectrum))
}
```