Normalization of Power Spectral Densities.

Andrew J. Barbour

February 6, 2013

Abstract

A vast and deep pool of literature exists on the subject of spectral analysis; wading through it can obscure even the most fundamental concepts to the inexperienced practitioner. Appropriate interpretation of spectral analyses depends crucially on the normalization used, and here we outline the normalization used by `rlpSpec`, namely the **single-sided power spectral density** (PSD). We briefly outline the background mathematics, present an example from scratch, and compare the results with the normalization used by the spectrum estimator included in the base distribution of R: `stats::spectrum`.

Contents

1 Background

There can often be confusion about the different quantities used in spectral analysis[1], partly due to myriad nomenclature within the incredibly vast literature on the subject. Commonly

[1] A nice illustration of the type of confusion common in spectral analyses of confusion is found in this thread on `R-help`:
`http://r.789695.n4.nabble.com/Re-How-do-I-normalize-a-PSD-td792902.html`

one finds similarly sounding phrases, including "amplitude spectrum", "energy spectral density", "power", "power spectra", and even "spectra". These all mean *something*, but are rarely equivalent, and can be used improperly.

To clarify these terms, we will walk through some background and definitions, without overly complicating the discussion with proofs. Let us first define a single realization of a stochastic process in the time domain, $\mathcal{X}(t)$, sampled over a finite interval of time $(-T/2, T/2)$, denoted by $X_T(t)$. A realization of $\mathcal{X}(t)$ will have an ordinary Fourier transform:

$$\tilde{X}_T(f) = \mathcal{F}\{X_T(t)\} = \int_{-\infty}^{\infty} X_T(t) e^{-2\pi i f t}\, dt = \int_{-T/2}^{T/2} \mathcal{X}(t) e^{-2\pi i f t}\, dt \tag{1}$$

The **amplitude spectrum** is the modulus of $\tilde{X}_T$ and the **phase spectrum** is the argument of $\tilde{X}_T$, although these are generally not informative for physical applications, if ever. The **energy spectral density** is found from $\tilde{X}_T$ by finding the expectation of the squared magnitude:

$$^{(\mathrm{E})}S(f) = \mathcal{E}\{|\tilde{X}_T(f)|^2\} \tag{2}$$

There is a problem though, in that as T grows to infinity, so too does $^{(\mathrm{E})}S(f)$. We divide it by the interval length T to tame such growth, which gives us an expression for **power spectral density**:

$$\begin{aligned} S(f) &= \lim_{T\to\infty} {}^{(\mathrm{E})}S(f)/T \\ &= \lim_{T\to\infty} \mathcal{E}\left\{ \frac{1}{T} \left| \int_{-T/2}^{T/2} \mathcal{X}(t) e^{-2\pi i f t}\, dt \right|^2 \right\} \end{aligned} \tag{3}$$

which is real, non-negative, and exists for all stationary processes $\mathcal{X}(t)$ with zero mean and finite variance.

Here is an important point to note regarding normalization. Equation (3) defines the **double-sided** PSD, because in it f runs from $\pm\infty$. If $\mathcal{X}(t)$ is real the power spectrum $S(f)$ is even; hence, we only need estimates for $f \geq 0$. The **single-sided** PSD is thus given by $2S(f)$ for $f \geq 0$. In many cases this sidedness distinction, as we will see, explains errant factors of two in PSD normalizations.

1.1 Connection to the autocovariance function

What is the connection between the PSD, defined in Equation (3), and the autocovariance function $\mathcal{R}(\tau)$?

From Equation (3) we see that $S(f)$ is obtained from products of $\mathcal{X}(t)$ with itself at any particular f, so it is related to the second-order moment of $\mathcal{X}(t)$ only; so too does the

autocovariance $\mathcal{R}(\tau)$. It may be surprising to note as well that $S(f)$ is simply the Fourier transform of $\mathcal{R}(\tau)$:

$$S(f) = \mathcal{F}\{\mathcal{R}(t)\} = \int_{-\infty}^{\infty} \mathcal{R}(t) e^{-2\pi i f t}\, dt \tag{4}$$

So, the functions $\mathcal{R}(\tau)$ and $S(f)$ can contain the same information. For real data, $\mathcal{R}(\tau)$ is always even, and always real. This implies that $S(f)$ is also a real and even function in f, which, because $S(f) >= 0$, restricts the functions $\mathcal{R}(t)$ could possibly represent. Put another way, there are many examples of even functions having non-positive Fourier transforms (see Bracewell (2000)).

1.2 Testing normalization

We can use a property of the autocovariance function $\mathcal{R}(\tau)$ to test whether or not a PSD is properly normalized. To see this, we take the inverse Fourier transform of Equation (4):

$$\mathcal{R}(t) = \int_{-\infty}^{\infty} S(f) e^{2\pi i f t}\, df \tag{5}$$

and recall a property of the autocovariance of a zero-mean process for zero lag:

$$\mathcal{R}(0) = \mathcal{E}\{\mathcal{X}(t)\mathcal{X}(t)\} = \mathcal{V}\{\mathcal{X}(t)\} = \sigma^2 \tag{6}$$

Using (6) and setting $t = 0$ in (4) gives the basis of our normalization test:

$$\sigma^2 = \int_{-\infty}^{\infty} S(f)\, df \tag{7}$$

That is, the area under the power spectrum is the variance of the process. So, a straightforward way to test normalization is to compute the PSD for a realization of $\mathcal{X}(t)$ with known variance and zero mean [e.g. $\mathcal{N}(0, \sigma^2)$], and calculate the integrated spectrum. For example, the single-sided PSD for a realization of a $\mathcal{N}(0, 1)$ process, sampled at 1 Hz, will be flat at 2 units^2/Hz across the entire band [0, 1/2], and will have an area equal to one.

2 A from-scratch example: White noise.

First, generate a series, and then find its Discrete Fourier Transform (DFT)[2].

[2] A proper DFT is normalized by the length of the series; however, most DFT calculators (including `stats::fft`) eschew this normalization for efficiency's sake.

```
set.seed(1234)
N <- 256
x <- rnorm(N, mean = 0, sd = 1)
xv <- var(x)
X <- fft(x)
class(X)

## [1] "complex"

length(X)

## [1] 256
```

We can easily find the amplitude and phase response:

```
Sa <- Mod(X)  # Amplitude spectrum
Sp <- Arg(X)  # Phase spectrum
```

followed by equivalent energy spectral density calculations[3]

```
XC <- Conj(X)
all.equal(Se <- Sa^2, Se_2 <- Mod(XC * X), Se_2R <- Mod(X * XC))

## [1] TRUE
```

The single-sided power spectral density (PSD) estimates follow once the Nyquist frequency is set; this is defined as half the sampling rate[4].

```
fsamp <- 1  # sampling freq, Hz
fNyq <- fsamp/2  # nyquist
Nf <- N/2
nyfreqs <- seq.int(from = 0, to = fNyq, length.out = Nf)
S <- Se[1:Nf] * 2/N  # Finally, the PSD!
print(c(mSn <- mean(S), mSm <- median(S)))

## [1] 2.034 1.242
```

[3] Note the equivalence between the complex conjugate based estimates.

[4] Although a white noise process is not strictly bandlimited, we will use it to demonstrate differences in normalization.

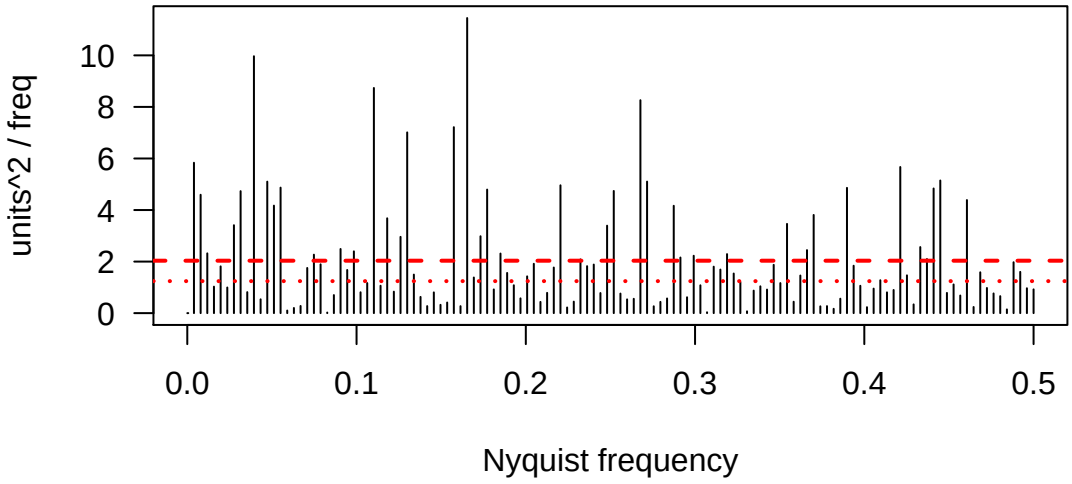

Figure 1: Power spectral density estimates for a single realization of a $\mathcal{N}(0,1)$ process, in linear units. The dashed line shows the mean spectral level and the dotted line shows the median spectral level; these can be used to find the integrated spectrum and test normalization.

An estimate of the integrated spectrum should roughly equal the known variance. Figure 1 plots the PSD of our white noise series and the mean and median values[5], from which we can perform a variance–normalization test:

```
test_norm <- function(sval, nyq, xvar) {
    svar <- sval * nyq
    return(svar/xvar)
}
print(xv_1 <- test_norm(mSn, fNyq, xv))

## [1] 0.9933

xv_2 <- sum(S)/Nf * fNyq/xv  # an alternate test
```

[5] Estimates for the PSD of a white noise series are approximately log-normally distributed; hence, a simple mean value tends to be highly biased estimator of expectation.

```
all.equal(xv_1, xv_2)

## [1] TRUE
```

But what if the sampling frequency `fsamp` changes? An obvious change will be the actual Nyquist frequency, which means the variance–normalization test will fail if the PSD estimates are not re-scaled. We simply re-scale the frequencies and PSD with the sampling rate to obtain the properly-normalized spectra.

```
fsamp <- 20
fNyq <- fsamp/2
freqs <- fsamp * nyfreqs
Snew <- S/fsamp
# Test variance crudely
mSn <- mean(Snew)
test_norm(mSn, fNyq, xv)

## [1] 0.9933
```

In Figure 2 we plot the PSD with new normalization, and compare it to the previous normalization. Spectral values are shown as decibels (relative to 1 units^2/frequency), using:

```
# decibel function
dB <- function(y) 10 * log10(y)
```

3 Normalization used in `stats::spectrum`

The PSD estimator included in the core distribution of R is `stats::spectrum`, which calls either `stats::spec.ar` or `stats::spec.pgram` for cases of parametric and non-parametric estimation, respectively. For this discussion we compare to `spec.pgram`; the user can, optionally, apply a single cosine taper, and/or a smoothing kernel.

By default `spec.pgram` assumes the sampling frequency for the input series is 1, and normalizes accordingly; however, the sampling information may be specified by creating a `ts` object from the series prior to spectrum estimation:

```
fsamp <- 20
xt <- ts(x, frequency = fsamp)
pgram20 <- spec.pgram(xt, pad = 1, taper = 0, plot = FALSE)
pgram01 <- spec.pgram(ts(xt, frequency = 1), pad = 1, taper = 0, plot = FALSE)
```

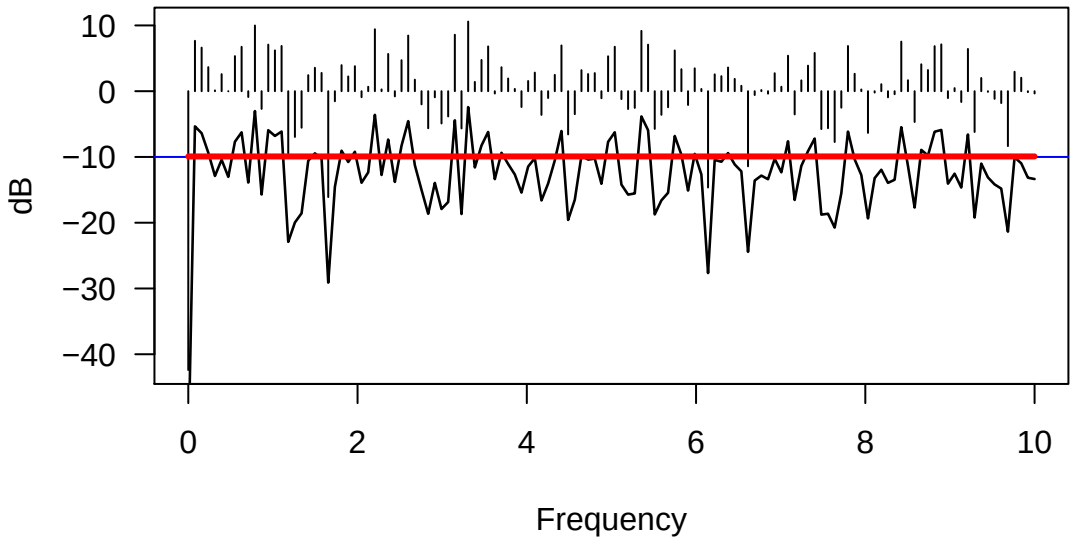

Figure 2: Rescaled PSD estimates for a single realization of a $\mathcal{N}(0,1)$ process with a sampling rate of 20 s^{-1} rather than 1 s^{-1} as from before. The thick red line shows the mean (rescaled) spectral level, and the blue line shows the predicted mean value based on twice the sampling frequency.

We plot the two PSD estimates on the same scales, in Figure 3, utilizing the plot method for `spec` objects: `plot.spec`. We also show horizontal lines corresponding to the inverse of twice the sampling rate, which puts the spectra about a factor of 2 too low:

```
mSn/mean(pgram20$spec)

## [1] 2.05
```

Because the frequencies are clearly correct, this factor of two likely means the spectra will fail our simple variance-normalization test. They do fail, by a factor of two, again too low:

```
test_norm(mean(pgram01$spec), 0.5, xv)

## [1] 0.4845
```

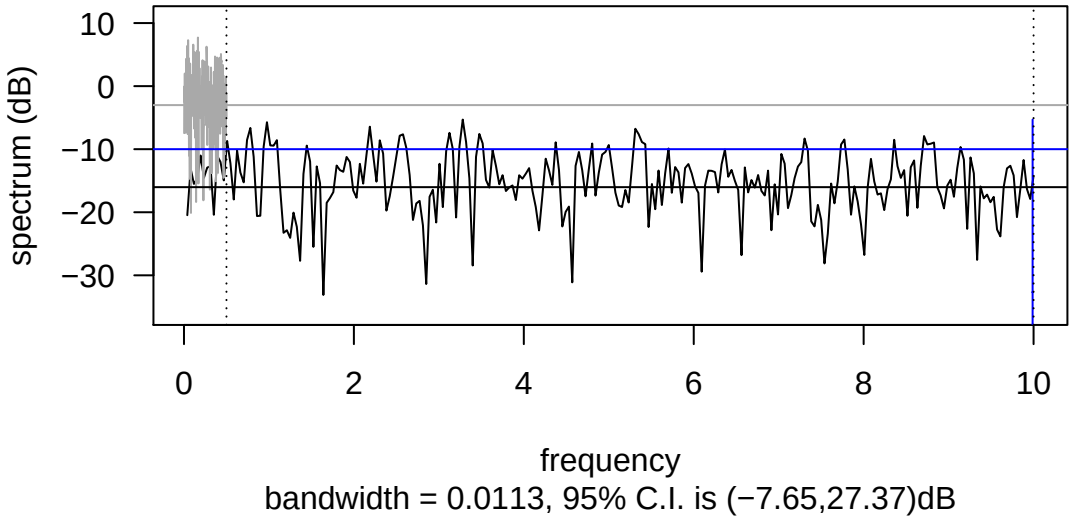

Figure 3: Power spectral densities from `spec.pgram` for the same data series. The grey curves show the PSD for a sampling rate of 1; whereas, the black curves show the PSD for a sampling rate of 20. The horizontal lines show levels corresponding to the inverse of twice the sampling rate (black and grey), and the expected spectral level for the 20 Hz sampling in blue. Vertical lines show the respective Nyquist frequencies.

```
test_norm(mean(pgram20$spec), 10, xv)

## [1] 0.4845
```

But why? This errant factor of two comes from the assumption of a double-sided spectrum, which is at odds with our definition of the single-sided spectrum by–you guessed it–a factor of two. We can illustrate this with the following example, where we compare the PSDs from `spec.pgram` for a real and complex series:

```
psd1 <- spec.pgram(x, plot = FALSE)
psd2 <- spec.pgram(xc <- complex(real = x, imag = x), plot = FALSE, demean = TRUE)
mx <- mean(Mod(x))
```

```
mxc <- mean(Mod(xc))
(mxc/mx)^2

## [1] 2

mean(psd2$spec/psd1$spec)

## [1] 2
```

Again, a factor of two. This means that unless we are interested in analyzing complex timeseries, we need only multiply by two to obtain properly normalized spectra from `spectrum`, assuming the sampling information is included in the series.

4 Other PSD estimators

The suite of extensions having similar functionality to `rlpSpec` is relatively limited; however, there are at least three which can produce sophisticated PSD estimates. We have summarized the available functions in Table 1 so far as we know[6].

Table 1: A comparison of power spectral density estimators in R, excluding extensions which only estimate raw-periodograms. Normalizations are shown as either "single" or "double" for either single- or double-sided spectra, and "various" if there are multiple, optional normalizations. A $(*)$ denotes the default for a function having an option for either single or double.

Function	Namespace	Sine m.t.?	Adaptive?	Norm.	Reference
`mtapspec`	`RSEIS`	Yes	No	various	Lees and Park (1995)
`pspectrum`	`rlpSpec`	Yes	Yes	single	Parker and Barbour (2013)
`spectrum`	`stats`	No	No	double	R Core Team (2012)
`spec.mtm`	`multitaper`	Yes	Yes	double	Rahim and Burr (2012)
`SDF`	`sapa`	Yes	No	single*	Percival and Walden (1993)

[6] As of this writing (Feb 2013), `sapa` appears to be orphaned.

References

Bracewell, R. (2000). *The Fourier Transform and its applications*. McGraw-Hill Science/Engineering/Math, 3 edition.

Lees, J. M. and Park, J. (1995). Multiple-taper spectral analysis: A stand-alone C-subroutine. *Computers & Geosciences*, 21(2):199–236.

Parker, R. L. and Barbour, A. J. (2013). *rlpSpec: Adaptive, sine-multitaper power spectral density estimation*. R package version 0.2-0.

Percival, D. and Walden, A. (1993). *Spectral analysis for physical applications*. Cambridge University Press.

R Core Team (2012). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Rahim, K. and Burr, W. (2012). *multitaper: Multitaper Spectral Analysis*. R package version 1.0-2.

Index