power spectral density

Normalization used in `rlpSpec`

Andrew J. Barbour

January 29, 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 **power spectral density** (PSD). We compare the PSD to other quantities commonly encountered in spectral analysis research, as well as normalizations used in a subset of the current suite of spectral analysis tools available in R.

Contents

1 Background **1**

2 A from-scratch example: White noise. **2**

3 Comparisons between estimators **5**

3.1 `stats::spectrum` 5

3.2 `stats::spectrum` 6

3.3 `multitaper::spec.mtm` 8

3.4 `SDF::sapa` 8

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 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.

[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`

Let us, for the sake of brevity, assume we are in the time domain, and we are considering a discrete stationary signal x of length N, having a Discrete Fourier Transform $\mathfrak{F}\{f\}$ represented by$X$. The **amplitude spectrum** of this transform pair is simply the amplitude of X, or $\mathrm{mod}\{X\}$, which we will denote this as ${}^{(\mathrm{A})}s$. This transform pair's corresponding **phase spectrum** is the phase angle of X, or $\arg\{X\}$, denoted by ${}^{(\phi)}s$.

amplitude spectrum

phase spectrum

power spectral density

How do we interpret the quantities

Table with equivalent expressions

Table 1: Representors of various spectral quantities.

Expression	Representing	Equivalent expressions
x	stationary timeseries	
X	Fourier transform of x	$\mathfrak{F}\{x\}$
${}^{(\mathrm{A})}s$	**amplitude spectrum** of x	$\lvert X\rvert$ or $\mathrm{mod}\{X\}$
${}^{(\phi)}s$	**phase spectrum** of x	$\arg\{X\}$
${}^{(\mathrm{E})}S$	**energy spectral density** of x	$\lvert X\rvert^2 = \mathrm{mod}\{X \star X^*\} = \mathrm{mod}\{X^* \star X\}$
${}^{(\mathrm{P})}S$	**power spectral density** of x	$\mathcal{E}\{\lvert X\rvert^2\} = \lvert X\rvert^2/N$

2 A from-scratch example: White noise.

A straightforward way to understand normalization in spectral analysis is to analyze a real, stationary series which is normally distributed with known variance, $x = \mathcal{N}(\mu, \sigma^2)$. A fundamental result found in many texts on spectral analysis is

$$\mathrm{var}\{x\} \equiv \sigma_x^2 = \int_{-1/2}^{1/2} {}^{(\mathrm{P})}S(f)df = 2\int_0^{1/2} {}^{(\mathrm{P})}S(f)df \tag{1}$$

which says if we integrate the power spectral density over all frequencies we can obtain the variance of the source process. If we have a $\mathcal{N}(0,1)$ process, and assume the sampling interval is once per second, we should expect a flat spectrum of 2 units^2/Nyquist across all frequencies $[0, 0.5]$ so that the area under the spectrum is equal to one.

We can illustrate this with a few lines of code. First, generate a series, and then find its iscrete Fourier Transform (DFT)[2].

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

[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.

```
[1] "complex"

> length(X)

[1] 256
```

energy spectral density
power spectral density

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!
```

An estimate of the integrated spectrum should roughly equal the known variance. Figure 2 plots the PSD of our white noise series; it also shows the mean value of the PSD[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] 0.9933334

> xv_2 <- sum(S)/Nf * fNyq / xv  # an alternate test
> all.equal(xv_1, xv_2)

[1] TRUE
```

[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.

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

```
> plot(nyfreqs, S, type="h", xlab="Nyquist frequency", ylab="units**2 / freq")
> print(c(mSn <- mean(S), mSm <- median(S)))

[1] 2.034495 1.242094

> abline(h=c(mSn,mSm), lwd=2, lty=c(2,3), col="red")
```

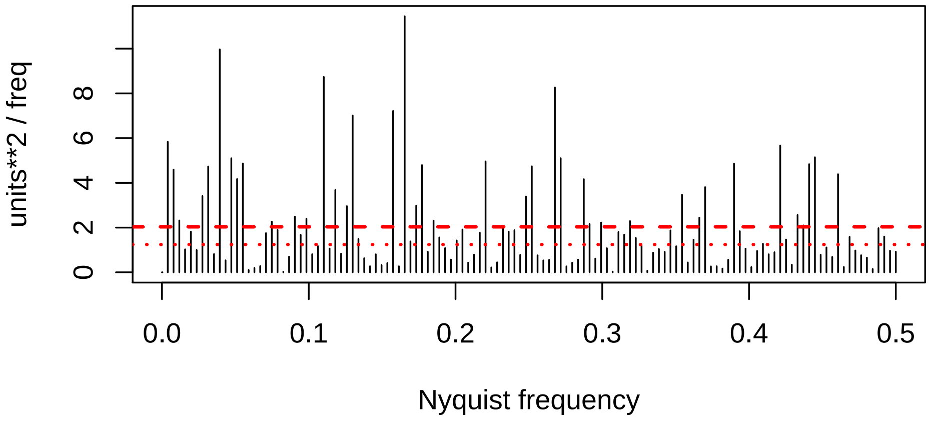

Figure 1: Power spectral density estimates for a single realization of a $\mathcal{N}(0, 1)$ process in linear units. The dashed line shows the mean spectral level and the dotted line shows the median spectral level; these can be used to find the integrated spectrum and test normalization.

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
```

To compare the scalings it is helpful to instead show the spectral values in decibels (relative to 1 units^2/frequency).

```
>     # decibel function
> dB <- function(y) 10*log10(y)
>     # and some plots...
> plot(freqs, dB(S), type="h", xlab="Frequency", ylab="dB")
> lines(freqs, dB(Snew), col="blue", lwd=2)
> abline(h=dB(1/fNyq), col="grey", lwd=2)
> mSn <- mean(Snew)
> lines(c(0,fNyq), rep(dB(mSn),2), lwd=2, lty=2, col="red")
>     # finally, test variance.
> test_norm(mSn, fNyq, xv)

[1] 0.9933334
```

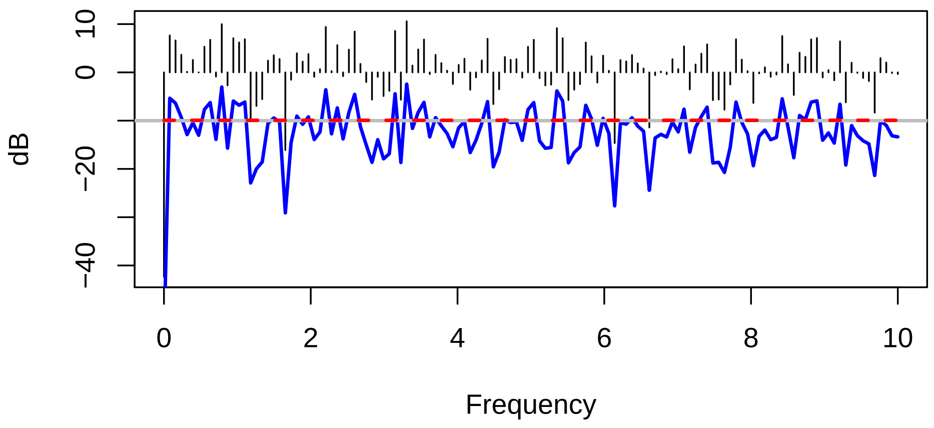

Figure 2: Rescaled PSD estimates for a single realization of a $\mathcal{N}(0,1)$ process with a sampling rate of 20 s^{-1} rather than 1 s^{-1} as from before. The dashed line shows the mean (rescaled) spectral level, and the grey line shows the predicted mean value from the Nyquist frequency.

3 Comparisons between estimators

We wish to compare the normalizations used by other PSD estimation programs; these are summarized in 2.

3.1 `stats::spectrum`

`spectrum` is a wrapper which calls the appropriate spectrum estimator. For our examples we compare to `spec.pgram`. Using default settings `spec.pgram` assumes the sampling frequency

Table 2: A comparison of functions comparable to `rlpSpec`, excluding extensions which only estimate raw-periodograms.

Function	Namespace	Sine m.t.?	Adaptive?	Reference
`mtapspec`	`RSEIS`	Yes	No	Lees and Park (1995)
`spectrum`	`stats`	No	No	R Core Team (2012)
`spec.mtm`	`multitaper`	Yes	Yes	Rahim and Burr (2012)
`SDF`	`sapa`	Yes	No	Percival and Walden (1993)

for the input series is 1, and normalizes accordingly. Sampling information used be included 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)
```

The first order question is obviously whether these spectra pass variance-normalization tests, which they do not, but only by a factor of two (too small):

But why? The program assumes normalization for complex series: Spectra returned for real data are deficient by a factor of two. Consider the following example:

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

[1] 1.116888

> mean((psd2$spec / psd1$spec))

[1] 2
```

This means that unless we are interested in analyzing complex timeseries, we need only multiply by two for properly normalized spectra using `spectrum`, assuming the sampling information is included in the input series.

3.2 `stats::spectrum`

Included in the core distribution of R is `stats::spectrum`, which accesses `stats::spec.ar` or `stats::spec.pgram` for either parametric and non-parametric estimation, respectively. The user can optionally apply a single cosine taper, and/or a smoothing kernel. Our method is non-parametric; hence, we will compare to the latter.

```
> spec.pgram(X, pad=1, taper=0.2, detrend=FALSE, demean=FALSE, plot=FALSE)
> test_norm(mean(pgram01$spec), 0.5, xv)

[1] 0.4845341

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

[1] 0.4845341

> plot(pgram20, log="dB", ylim=36*c(-1,.3))
> plot(pgram01, log="dB", add=TRUE, col="red")
> abline(h=dB(c(1, 1/2/1, 1/2/20)), col=c("grey","red","black"))
```

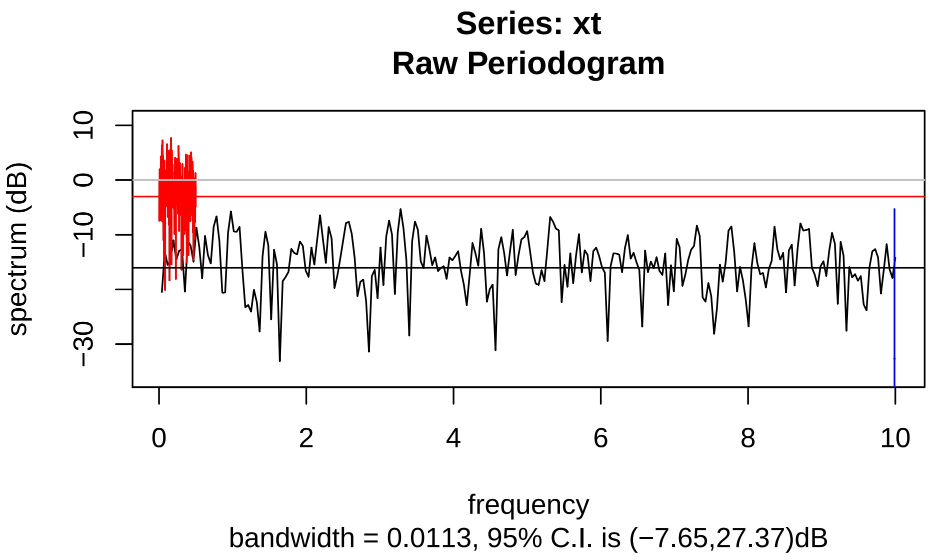

Figure 3: `spec.pgram`

However, the logical arguments `detrend` and `demean` to `psdcore` are passed to `spec.pgram`; they are, by default, both True.

As a matter of bookkeeping, we must deal with the working environment accessed by `rlpSpec` functions. Specifically, we should ensure `psdcore` does not access any inappropriate information by setting `refresh=TRUE`. We can then re-calculate the multitaper PSD and the raw periodogram with `plotpsd=TRUE`. The results are shown in Figure ??.

3.3 `multitaper::spec.mtm`

3.4 `SDF::sapa`

References

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

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

amplitude spectrum, 1, 2

energy spectral density, 2, 3

phase spectrum, 1, 2
power spectral density, 1–3