amplitude spectrum
phase spectrum

Normalization of Spectral Analyses

Andrew J. Barbour

January 28, 2013

Abstract

Units matter, and having correct ones is crucial to interpretation of spectral analyses. Here we outline the normalization used by `rlpSpec`, namely the power spectral density, and compare it to other quantities commonly encountered in spectral analysis.

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. Regarding nomenclature: Phrases including "amplitude spectrum", "energy spectral density", "power", "power spectra", and even "spectra" all mean *something*, but are rarely equivalent.

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

How do we interpret the quantities

Table with equivalent expressions

[1] This post to `R-help` illustrates the confusion common in spectral analyses:
`http://r.789695.n4.nabble.com/Re-How-do-I-normalize-a-PSD-td792902.html`

Table 1: Representors of various spectral quantities.

Expression	Representing	Equivalent expression
f	stationary timeseries	
F	Fourier transform of f	$\mathfrak{F}\{f\}$
$S^{(\mathrm{A})}$	**amplitude spectrum** of F	$\lvert F\rvert$ or $\mathrm{mod}\{F\}$
$S^{(\phi)}$	**phase spectrum** of F	$\arg\{F\}$
S	**energy spectral density** of $S^{(\mathrm{A})}$	
S	**power spectral density** of $S^{(\mathrm{A})}$	

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

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

which says if we integrate the power spectral density over all frequencies we can obtain the process variance. 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 first generate a series, and find its Fourier transform[2].

```
> 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 immediately find the amplitude and phase response:

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

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

followed by the energy spectral densities[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 we set the Nyquist frequency, 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!
```

```
> plot(nyfreqs, S, type="h", xlab="Nyquist frequency", ylab="units**2 / freq")
> mSn <- mean(S)
> polygon(c(0,fNyq,fNyq,0,0), c(0,0,mSn,mSn,0), lwd=2, lty=2, border="red")
```

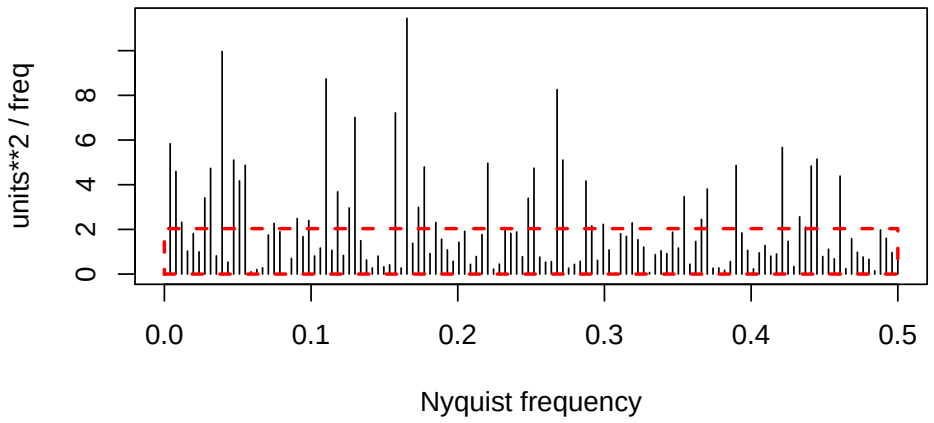


Figure 1: Power spectral density estimates for a single realization of a $\mathcal{N}(0,1)$ process in linear units. The dashed rectangle encompasses the area under the mean spectral level, used to find the integrated spectrum.

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

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

Figure 2 plots the PSD of our white noise series; it also shows the mean value of the PSD, from which we can test ourselves for proper normalization: an estimate of the integrated spectrum should roughly equal the known variance.

```
> 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
```

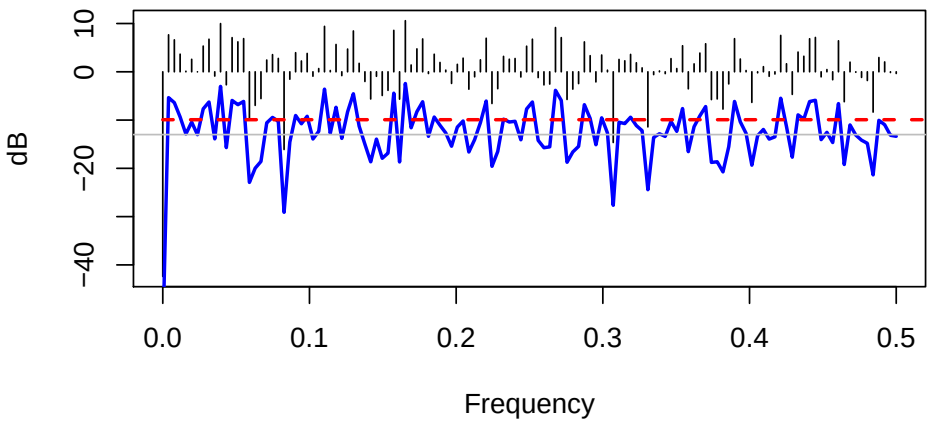
But what if the sampling frequency `fsamp` changes? An obvious change will be the actual Nyquist frequency, which means the variance normalization-test will fail if the PSD estimates are not re-scaled. We simply re-scale the frequencies and PSDs with the sampling rate to obtain the properly-normalized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

```
> fsamp <- 20
> fNyq <- fsamp/2
> freq <- fsamp*nyfreqs
> Snew <- S / fsamp
```

To compare the scalings it will be useful to show the spectral values in decibels (relative to units^2/frequency):

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

```
[1] 0.9933334
```



3 Comparisons between estimators

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

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)

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 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):

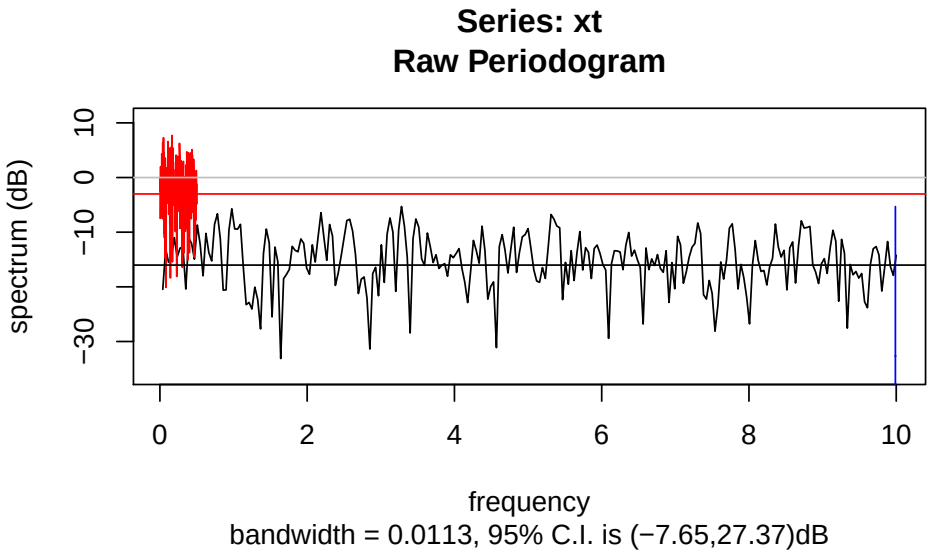
```
> test_norm(mean(pgram01$spec), 0.5, xv)

[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"))
```



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)
```

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