Normalization of Power Spectral Density estimates.

Andrew J. Barbour and Robert L. Parker

February 25, 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 from the inexperienced practitioner. Appropriate interpretation of spectral analyses depends crucially on the normalization used, which is often the greatest source of confusion. Here we outline the normalization used by `psd`, 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 Introduction

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

[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`

similarly sounding phrases, including "amplitude spectrum", "energy spectral density", "power", "power spectra", and even "spectra". These all mean *something*, but are rarely equivalent, and are sometimes used improperly. To clarify these terms, we will walk through some background and definitions, without overly complicating the discussion with proofs.

1.1 Power spectral density: An informal definition

We begin by considering a stationary stochastic process $\mathcal{X}(t)$, a random function extending throughout all time with time-invariant properties. Our goal is to characterize $\mathcal{X}(t)$ with an ordinary function describing its properties in frequency (as the autocorrelation function does in time).

Functions of a real variable (continuous time signals) will have a Fourier transform

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

and for discrete-time signals $\mathcal{X}_n$ the spectrum is restricted to the finite interval

$$\tilde{\mathcal{X}}(f) = \sum_{n=-\infty}^{\infty} \mathcal{X}_n e^{-2\pi i f n}, -1/2 \leq f \leq 1/2 \tag{2}$$

which means we can reconstruct the original series from the inverse transform of (2), over frequencies $(-1/2,\ 1/2)$:

$$\mathcal{X}_n = \mathcal{F}^{-1}\{\tilde{\mathcal{X}}(f)\} = \int_{-1/2}^{1/2} \mathcal{X}(t) e^{2\pi i f n}\, df \tag{3}$$

1.2 The filter analogy

There is a problem with the definitions we have just presented. The integral in (1) or the sum in (2) must converge to apply these equations to their corresponding stochastic processes; this would require some kind of decrease in amplitude, or energy, as t or n gets large which does not happen for a stationary process. And, if we put a stochastic process in $f(t)$ in (1) we would obtain another random function, which is contrary to our stated goal of obtaining a descriptive function in frequency space.

Suppose we wanted to know how much variability the stationary process exhibits at a frequency f_0. We could design a narrow bandpass filter ϕ_f that only allowed signals through in the frequency band $(f_0 - 1/2\Delta f, f_0 + 1/2\Delta f)$ with unit gain. If we convolve a signal X with that filter to obtain the process X_ϕ, we will have a stochastic process having very limited frequency content. We would expect the variance (a measure of amplitude) to be proportional to the width Δf of the bandpass filter, which prevents the condition of infinite energy at infinite time (or length). The variance of the filtered process X_ϕ will be some positive value times Δf, and will vary as the center frequency f_0 is varied. At the frequency f_0 the following is also true:

$$\mathcal{V}\{X_\phi\} \propto \mathcal{V}\{X\}$$

The variance of X in a frequency band is called the **power** in that band, and so S_X is the power spectrum of X, or more grandly its **power spectral density**:

$$S_X(f_0)\Delta f = \mathcal{V}\{\phi_f \star X\} \tag{4}$$

Equation (4) is our informal definition of $S_X(f_0)$. Notice this definition works equally well for continuous or discrete processes. In the days before computers, analog spectral analyzers were built based on this principle: a large number of narrow bandpass filters followed by rectifiers to measure the variance in each band.

We can demonstrate this analogy by plotting the PSDs obtained for a normally distributed process, and a bandpass filtered version of it, along with the filter's amplitude response. In Figure 1 we see that within the pass-band the variance of the filtered process is equal to that of the infinitely long process. We could imagine the grey curve being comprised of a collection of passband responses. If we could construct a true bandpass filter, there would be zero energy outside the passband; but, in practice the filter weights create side-lobe energy which is unavoidable. Also note the artificial bias from spectral leakage, seen just outside the edges of the pass-band.

```
set.seed(1234)
N <- 1028
x <- rnorm(N, mean = 0, sd = 1)
# Construct an FIR filter
library(signal)

## Loading required package:  MASS
##
## Attaching package:  'signal'
## The following object(s) are masked from 'package:stats':
##
## filter, poly

f <- c(0, 0.2, 0.2, 0.3, 0.3, 0.5) * 2
m <- c(0, 0, 1, 1, 0, 0)
fw <- signal::fir2(N, f, m)
# complex filter response
fh <- signal::freqz(fw, Fs = 1)
f <- c(0, 0.12, 0.12, 0.22, 0.22, 0.5) * 2
fwl <- signal::fir2(N, f, m)
fhl <- signal::freqz(fwl, Fs = 1)
f <- c(0, 0.28, 0.28, 0.38, 0.38, 0.5) * 2
fwu <- signal::fir2(N, f, m)
fhu <- signal::freqz(fwu, Fs = 1)
# convolution
xf <- signal::filter(fw, x)
```

```
# PSD using stats::spectrum
Sx <- spectrum(x, pad = 1, plot = FALSE, taper = 0.2)
Sxf <- spectrum(xf, pad = 1, plot = FALSE, taper = 0.2)
```

```
Sx$df <- Sx$bandwidth <- NA
par(mar = c(0, 0, 2, 0), oma = rep(0, 4))
plot(Sx, col = "grey", log = "dB", ylim = c(-150, 20), xlim = c(0.1, 0.4), xlab = "",
    ylab = "", ci.col = NA, axes = FALSE, main = "PSDs of raw and filtered process")
lines(fhu$f, 20 * log10(Mod(fhu$h)), col = "red", lty = 3, lwd = 2)
lines(fhl$f, 20 * log10(Mod(fhl$h)), col = "red", lty = 3, lwd = 2)
lines(fh$f, 20 * log10(Mod(fh$h)), col = "red", lwd = 2)
plot(Sxf, log = "dB", add = TRUE)
```

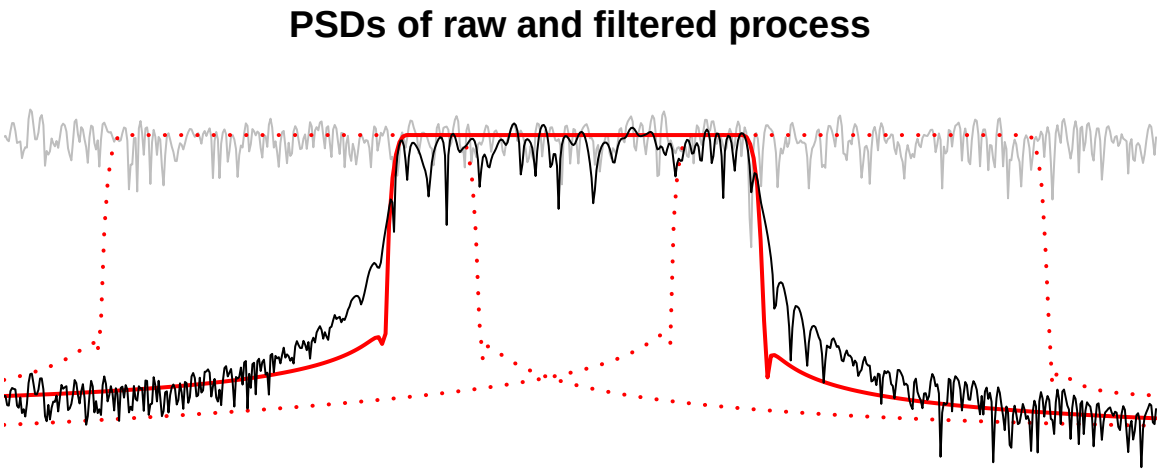

Figure 1: Demonstration of the "filter analogy". A normally-distributed process (with PSD in grey) is convolved with a bandpass filter (with a response shown in red); the resulting PSD is in black. In this analogy the PSD is the variance of a stationary process in an infinitesimally narrow band, as if we had passed our signal through a bank of narrow filters (dotted lines) and looked at the output from each one.

1.3 Power spectral density: A more formal definition

Let us return to having a single realization of a stochastic process in the time domain, $\mathcal{X}(t)$, which is sampled over a finite interval of time $(-T/2, T/2)$, denoted by $X_T(t)$. A realization of $\mathcal{X}(t)$ will have a Fourier transform:

$$\tilde{X}_T(f) = \mathcal{F}\{X_T(t)\} = \int_{-\infty}^{\infty} X_T(t) e^{-2\pi i f t}\, dt = \int_{-T/2}^{T/2} \mathcal{X}(t) e^{-2\pi i f t}\, dt \tag{5}$$

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 amplitude spectrum:

$$E(f) = \mathcal{E}\{|\tilde{X}_T(f)|^2\} \tag{6}$$

We note the necessity for convergence mentioned previously: as T grows to infinity, so too does $E(f)$. We divide it by the interval length T to curb this growth, which gives us an expression for **power spectral density**:

$$\begin{aligned} S(f) &= \lim_{T\to\infty} T^{-1} E(f) \\ &= \lim_{T\to\infty} \mathcal{E}\left\{ \frac{1}{T} \left| \int_{-T/2}^{T/2} \mathcal{X}(t) e^{-2\pi i f t}\, dt \right|^2 \right\} \end{aligned} \tag{7}$$

which is real, non-negative, and exists for all stationary processes with zero mean and finite variance.

Equation (7) defines the **double-sided** PSD, because in the limit of T, the limits of integration are $\pm\infty$. 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.

1.4 Connection to the autocovariance function

What is the connection between the PSD, defined in Equation (7), and the autocovariance function $\mathcal{R}(\tau)$?

From Equation (7) 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** (ACV) $\mathcal{R}(\tau)$:

$$\begin{aligned} \mathcal{R}(\tau) &= \mathrm{Cov}\left(\mathcal{X}(\tau), \mathcal{X}(\tau)\right) \\ &= \mathcal{E}\left\{ \left(\mathcal{X}(\tau) - \mathcal{E}\{\mathcal{X}(\tau)\}\right)^2 \right\} \end{aligned} \tag{8}$$

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

$$S(f) = \mathcal{F}\{\mathcal{R}(\tau)\} = \int_{-\infty}^{\infty} \mathcal{R}(\tau) e^{-2\pi i f \tau}\, d\tau \tag{9}$$

So, the functions $\mathcal{R}(\tau)$ and $S(f)$ exist as a transform pair. 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)).

1.5 Testing normalization

We can use a property of the ACV function $\mathcal{R}(\tau)$ or the informal definition in (**??**) to test whether or not a PSD is properly normalized. To see this, we take the inverse Fourier transform of (9):

$$\mathcal{R}(t) = \int_{-\infty}^{\infty} S(f) e^{2\pi i f t}\, df \tag{10}$$

and find the ACV of a zero-mean process for zero lag. From (8) we have:

$$\mathcal{R}(0) = \mathcal{E}\{\mathcal{X}(t)^2\} = \mathcal{V}\{\mathcal{X}(t)\} = \sigma_{\mathcal{X}}^2 \tag{11}$$

and by setting $t = 0$ in (10) we have the basis of our normalization test:

$$\sigma_{\mathcal{X}}^2 = \int_{-\infty}^{\infty} S(f)\, df \tag{12}$$

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 then 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 $2\ ^{\text{units}^2}/_{\text{Hz}}$ across the frequency band $[0, 1/2]$, and will have an area equal to one.

1.6 Summary of nomenclature

In Table 1 we give a summary of some of the quantities we have reviewed.

2 A from-scratch example: White noise.

Without using the tools in `psd` we will build up an example using R commands, in order to highlight the topic of normalization.

First, generate a normally distributed series[2], and then find its Discrete Fourier Transform (DFT)[3].

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

[3] 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.

Table 1: Top: Representers of various spectral quantities and their associated equations in the main text. Bottom: Normalizations for double and single sided PSD estimates to maintain process variance of $\sigma^2_{\mathcal{X}}$ and frequency ranges (f_N is the Nyquist frequency).

Expression	Representing	Equivalent	Equation
$\mathcal{X}(t)$	stochastic process with variance $\sigma^2_{\mathcal{X}}$		
$X_T(t)$	single realization of $\mathcal{X}(t)$ over T		
$\mathcal{F}\{X_T(t)\}$	Fourier transform of $X_T(t)$	$\tilde{X}_T(f)$	5
$\sqrt{\Re^2 + \Im^2}$	**amplitude spectrum** of $X_T(t)$	$\lvert\tilde{X}_T\rvert$ or $\text{mod}\{\tilde{X}_T\}$	
$\tan^{-1}(\Im/\Re)$	**phase spectrum** of $X_T(t)$	$\arg\{\tilde{X}_T\}$	
$\lvert\tilde{X}_T\rvert^2$	**energy spectral density** of $X_T(t)$	$E(f)$ or $\text{mod}\{\tilde{X} \star \tilde{X}^*\}$	6
$\lvert\tilde{X}_T\rvert^2/N$	**power spectral density** of $X_T(t)$	$S(f)$ or $\mathcal{E}\{\lvert X\rvert^2\}$	7
$\mathcal{F}^{-1}\{S(f)\}$	**autocovariance** of $X_T(t)$	$\text{Cov}\,(X_T(t), X_T(t))$	8 and 9

Normalizations to maintain $\sigma^2_{\mathcal{X}}$		Frequency range
$S(f)$	**double-sided**	$[-f_N, f_N]$
$2\ S(f)$	**single-sided**	$[0, f_N]$

```
# using x from the filter analogy section
xv <- var(x)
X <- fft(x)
class(X)

## [1] "complex"

length(X)

## [1] 1028
```

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 estimates[4]:

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

```
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 estimates follow once we have the Nyquist frequency, defined as half the sampling rate:

```
fsamp <- 1  # sampling freq, e.g. Hz
fNyq <- fsamp/2  # Nyquist freq
Nf <- N/2  # number of freqs
nyfreqs <- seq.int(from = 0, to = fNyq, length.out = Nf)
S <- Se[1:Nf] * 2/N  # Finally, the PSD!
```

To approximate the integrated spectrum in the case of a "flat" spectrum, we need an accurate measure of the first moment of the spectral values. The `median` estimator produces a biased estimate because the distribution of spectral values roughly follows a χ^2_ν distribution, where ν is the number of degrees of freedom and, for this distribution, the expectation of the first moment. To find this value we perform a conjugate gradient based minimization of the best-fitting χ^2 distribution, and compare this with the value returned by `mean`. Our starting point will be the estimated mean value. We visualize the fit with a "Q-Q" plot, which shows PSD quantiles values as a function of χ^2 quantiles, using the optimized value of the number of degrees of freedom; this is shown in Figure 2.

```
# 0) Setup optimization function for dof, using conjugate gradients\\
# min L1 |PSD - Chi^2(dof)|
Chifit <- function(PSD) {
    optim(list(df = mean(PSD)), function(dof) {
        sum(log(PSD)) - sum(log(dchisq(PSD, dof)))
    }, method = "CG")
}
# 1) run optimization
Schi <- Chifit(S)
# Get 'df', the degrees of freedom
print(dof <- Schi$par[[1]])

## [1] 2.005
```

While the optimal degrees of freedom is very nearly two, the value produced by `mean` is different by merely one percent (certainly an acceptable bias for our purposes).

```
# compare with the mean and median
c(mSn <- mean(S), median(S))

## [1] 1.989 1.406
```

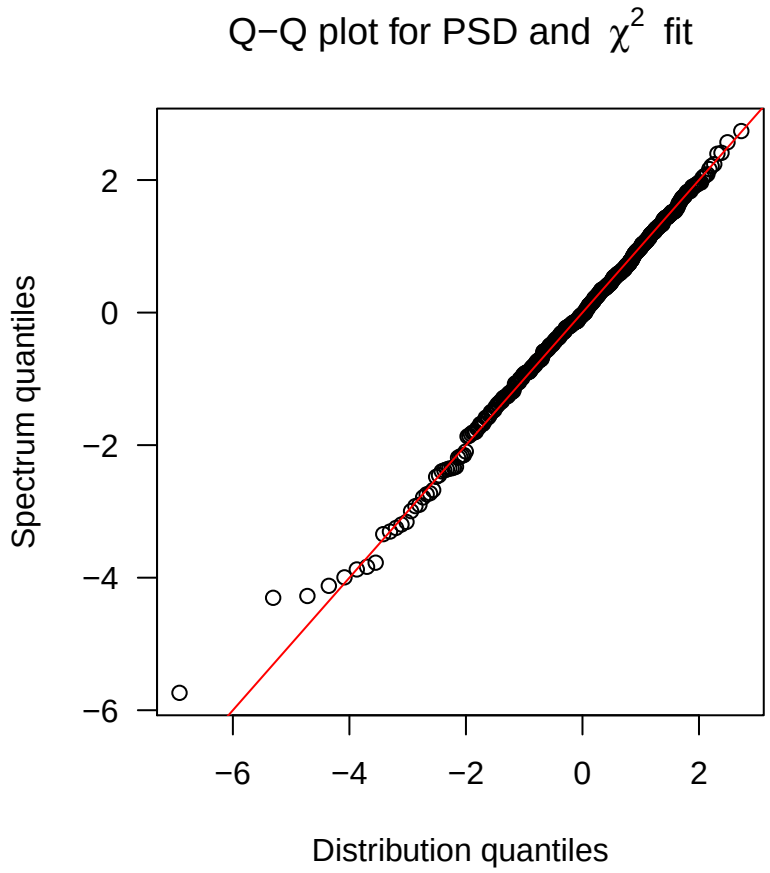

Figure 2: "Q-Q" plot of quantiles of our example PSD against theoretical χ^2_ν quantiles. The distribution is calculated with a value for degrees of freedom (ν) obtained from the L_1-norm minimization procedure.

An estimate of the integrated spectrum should roughly equal the known variance. Figure 3 plots the PSD of our white noise series with the value of ν from the optimization, with which we can perform a variance–normalization test:

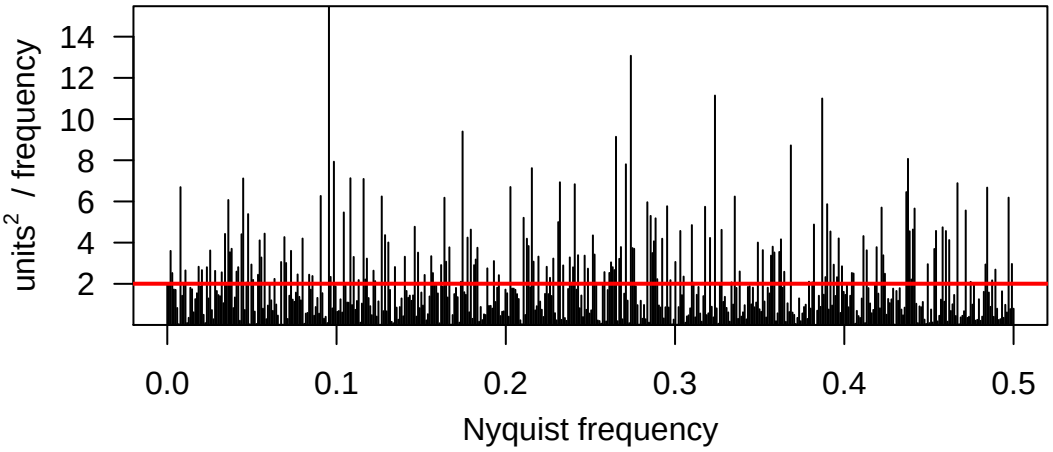

Figure 3: Power spectral density estimates for a single realization of a $\mathcal{N}(0,1)$ process, in linear units. The horizontal line shows the expectation of the spectral estimates, obtained from the χ^2 fit in Figure 2; this value is used to test normalization.

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

## [1] 1.009

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

## [1] "Mean relative difference: 0.007985"
```

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] 1.001
```

In Figure 4 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)
```

We plot the two PSD estimates on the same scales, in Figure 5, 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] 2.004
```

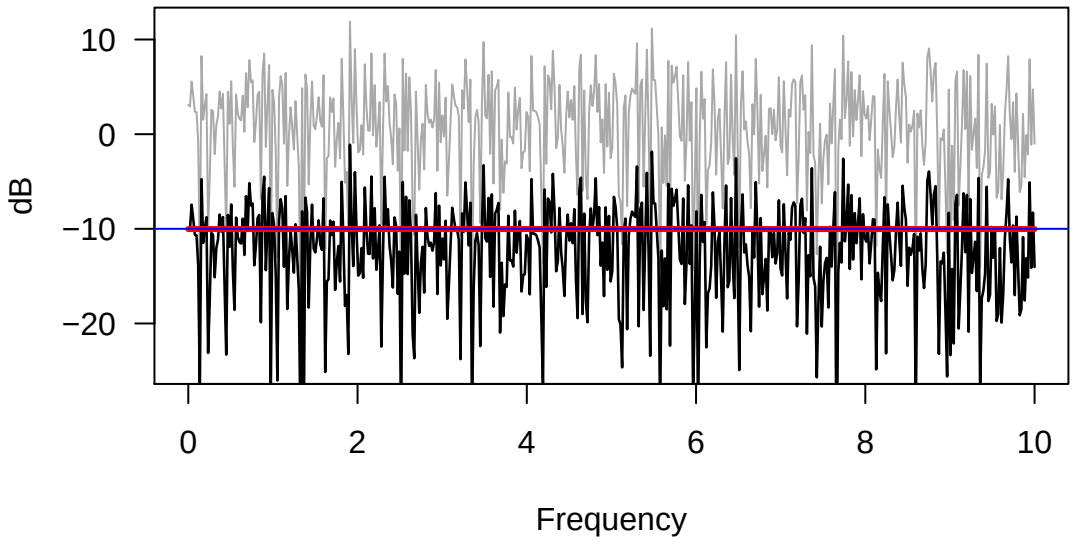

Figure 4: Rescaled PSD estimates for a single realization of a $\mathcal{N}(0,1)$ process with a sampling rate of 20 s^{-1} rather than 1 s^{-1} as from before. The original spectrum (grey) is scaled to the appropriate level (black). The thick red line shows the mean (rescaled) spectral level, and the blue line shows the predicted mean value based on twice the sampling frequency.

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] 0.4994

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

## [1] 0.4994
```

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`

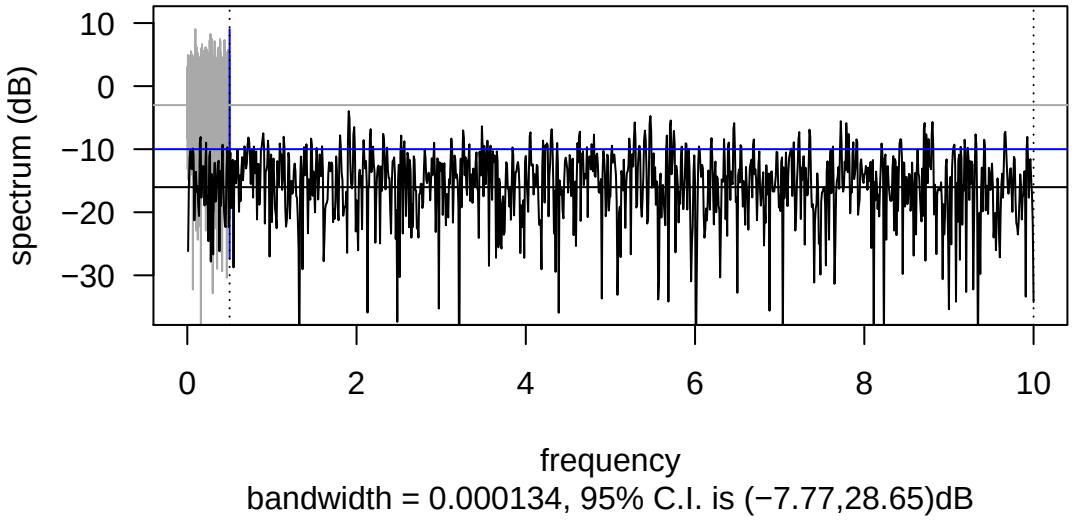

Figure 5: Power spectral densities from `spec.pgram` for the same data series. The grey series is the PSD for a sampling rate of 1; whereas, the black series is the PSD for a sampling rate of 20. The horizontal lines show levels corresponding to the inverse of twice the sampling rate (black and grey), and the expected spectral level for the 20 Hz sampling (blue). Vertical lines show the respective Nyquist frequencies.

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 PSD estimators in R

The suite of extensions having similar functionality to `psd` is relatively limited; however, there are at least four which can produce sophisticated PSD estimates. We have summarized the available functions in Table 2 so far as we know[5].

Table 2: 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
`bspec`	`bspec`	No	No	single*	Röver et al. (2011)
`mtapspec`	`RSEIS`	Yes	No	various	Lees and Park (1995)
`pspectrum`	`psd`	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)

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

Session Info

```
sessionInfo()

## R version 2.15.2 (2012-10-26)
## Platform: x86_64-apple-darwin9.8.0/x86_64 (64-bit)
##
## locale:
## [1] C
##
## attached base packages:
##  [1] parallel  datasets  grDevices grid      graphics  tools     stats
##  [8] utils     methods   base
##
## other attached packages:
## [1] signal_0.7-2 MASS_7.3-22  knitr_0.9
##
## loaded via a namespace (and not attached):
## [1] digest_0.6.0   evaluate_0.4.3 formatR_0.7    stringr_0.6.2
```

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). *psd: 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.

Röver, C., Meyer, R., and Christensen, N. (2011). Modelling coloured residual noise in gravitational-wave signal processing. *Classical and Quantum Gravity*, 28(1):015010.

Index