`rlpSpec`: Adaptive sine multitaper power spectral density estimation

Andrew J. Barbour and Robert L. Parker

January 30, 2013

Abstract

The purpose of this vignette is to provide an overview of the features included in `rlpSpec`, which allow the user to compute sophisticated power spectral density (PSD) estimates for a univariate series, with very little tuning effort. The sine multitapers are used in which the number of tapers varies with spectral shape, according to the optimal value proposed by Riedel and Sidorenko (1995). The adaptive procedure iteratively refines the optimal number of tapers at each frequency, which, assuming convergence, will produce spectra with significantly reduced variance (compared to naïve estimators), and minimum biasing effects. Resolution and uncertainty in a multitaper scheme are controlled by the number of tapers used. This means we do not need to resort to either windowing methods which inherently degrade resolution of low-frequency features (e.g. Welch), or smoothing kernels which can badly distort important features without careful tuning (e.g. Daniell, as in `stats::spec.pgram`). In this sense `rlpSpec` is best suited for data having spectra with both large dynamic range and strong, sharply changing features.

Contents

1 Quick start: A minimal example.

First load the package into the namespace:

```
> library(rlpSpec)
```

We now need a dataset to analyze. Among the datasets included in `rlpSpec` is a subset of the Magnetic Satellite (MAGSAT) mission (Langel et al., 1982). Specifically, we have included along-track measurements of horizontal magnetic-field strength from a gimballed, airborne magnetometer, sampled once every kilometer, which means the spectrum may represent crustal magnetization with wavelengths longer than 2 km.

```
> data(magsat)
```

The format of the data set is a `data.frame` with four sets of information:

```
> names(magsat)

[1] "km"    "raw"   "clean" "mdiff"
```

The `raw` and `clean` names represent raw and edited intensities respectively, expressed in units of nanoTesla; `mdiff` is the difference between them. The difference between them is a matter of just a few points attributable to instrumental malfunction.

```
> subset(magsat, abs(mdiff)>0)

    km    raw   clean     mdiff
403  0  209.1 -3.6355 -212.7355
717  0 -248.7 -9.7775  238.9225
```

These deviations can, as we will see, adversely affect the accuracy of any PSD estimate, multitaper or otherwise.

Setting aside any discussion regarding sample stationarity, we can find power spectral density (PSD) estimates for the two series quite simply:

```
> psdr <- pspectrum(magsat$raw)
> psdc <- pspectrum(magsat$clean)
```

Each `pspectrum` command calculates a pilot PSD, followed by `niter` iterations of refinement (defaulting to 3). With each iteration the number of tapers is adjusted to the optimal number, based on the weighted spectral derivatives, following Riedel and Sidorenko (1995). In general, spectral variance is reduced with sequential refinements[1], but is not necessarily guaranteed to converge. Note that in the example the sampling frequency of both series is km^{-1}, so we need not change the sampling rate argument.

Let's now visualize the two PSD estimates, recalling that the difference between the `raw` and `clean` samples is a mere two points. [2]

Figure 1 compares the spectra for the `raw` and `clean` samples. This plot shows a drastic improvement in shape between the two series, simply because the large outliers have been removed. The `clean` PSD shows the very red spectrum typical of geophysical processes (Agnew, 1992). It also shows a rolloff in signal for 10 kilometer wavelengths and longer; whereas, the `raw` PSD looks highly unrealistic at higher wavelengths, and shows some curvature bias.

2 Comparisons with other methods

As we have shown in the MAGSAT example, improved understanding of the physics behind the signals in the data is of great concern. Assuming a sample is free of non-physical points, how do PSD estimates from `rlpSpec` compare with other methods? Unfortunately the suite of extensions with similar functionality is relatively limited, but hopefully we have summarized most, if not all, the available functions in Table 1.

We now perform some tests to get a sense of how the results of `rlpSpec` compare with the methods in Table 1 for the same data, specifically the cleaned MAGSAT series.

2.1 `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.

Included in `rlpSpec` is an option to compare the results with a naïve estimator–a 20% tapered periodogram–from within the spectrum calculator, `psdcore`. In R this estimator is equivalent to running:

[1] Messages given by `pspectrum` with "Ave. S.V.R." are in reference to the average spectral-variance reduction, found from double-differenced spectra for each stage relative to the pilot estimate.

[2] Note that `pspectrum` returns an object with class `spec`, so we have access to methods within `stats`, including `plot.spec`.

```
> plot(psdc, log="dB", main="Raw and Clean MAGSAT power spectral density",
+        lwd=3, ci.col=NA, ylim=c(0,32), yaxs="i")
> plot(psdr, log="dB", add=TRUE, lwd=3, lty=5)
> legend("bottomleft", c("Raw","Clean"), title="Series", lwd=3, cex=1.1, lty=c(1,4))
```

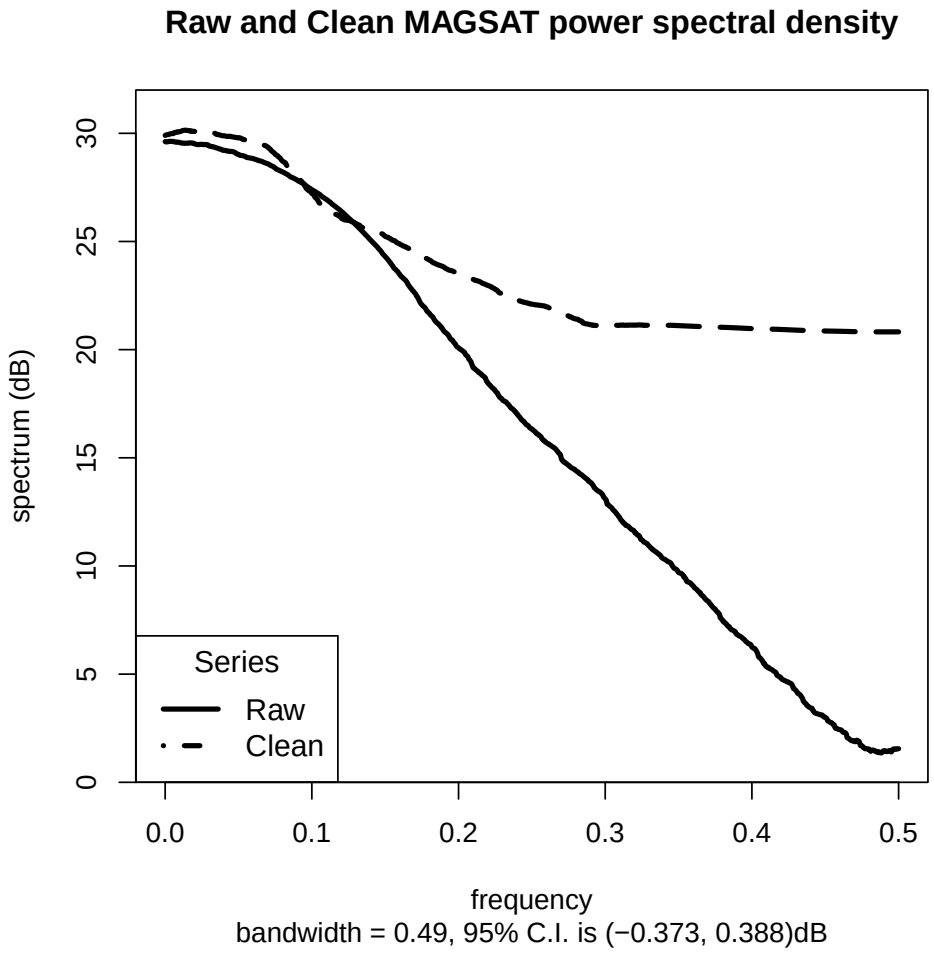


Figure 1: Power spectral density estimates for the raw and cleaned `MAGSAT` data included with `rlpSpec`.

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)

```
> spec.pgram(X, pad=1, taper=0.2, detrend=FALSE, demean=FALSE, plot=F)
```

except that in `psdcore` the logical arguments `detrend` and `demean` are passed to `spec.pgram` and are, by default, both TRUE.

As a matter of bookkeeping, we should 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 2.

```
> ntap <- psdc$taper
> psdcore(magsat$clean, ntaper=ntap, refresh=TRUE, plotpsd=TRUE)
```

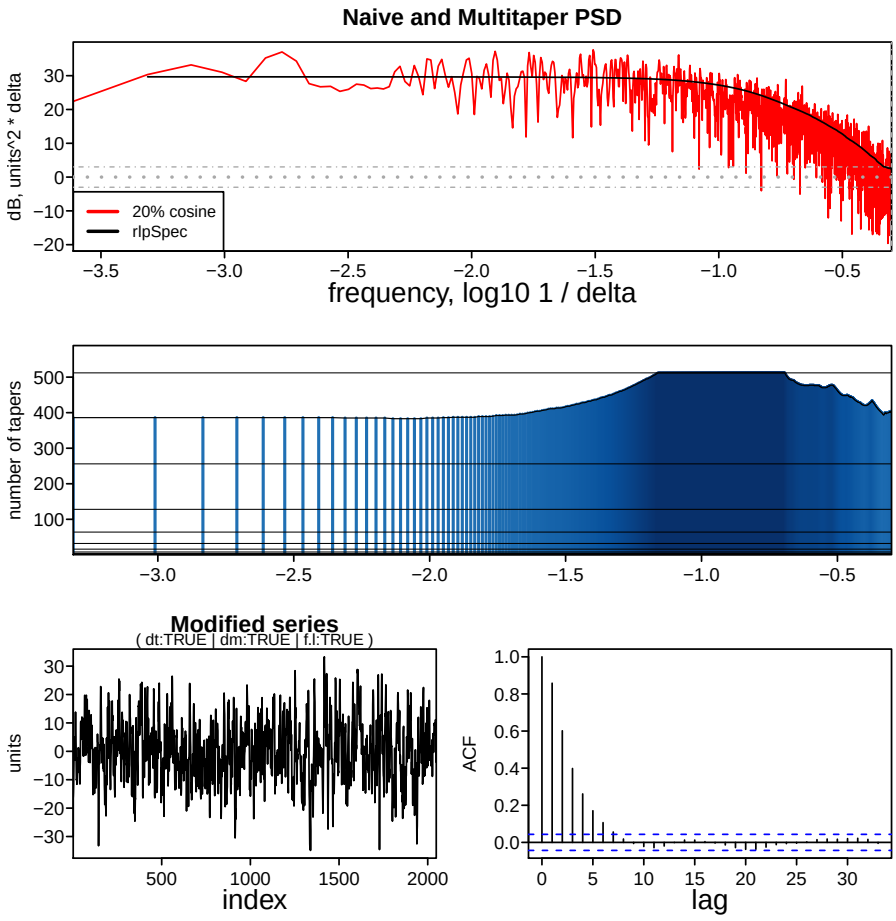


Figure 2: Top: Comparison between naïve and multitaper PSD estimators for the clean `MAGSAT` data. The frequency axis is in units of $\log_{10}$ km^{-1}, and power axis is in decibels. Middle: The number of tapers applied as a function of frequency from the `plot.tapers` method. Bottom: The spatial series used to estimate the PSDs and a subset of the full autocorrelation function.

2.2 RSEIS::mtapspec

The main spectrum estimation tool is `mtapspec`, which calls the program of Lees and Park (1995). With this program there are many tuning parameters to set, including normalization and taper averaging flags. For our purposes the correct normalization for `mtapspec` is found by using `MTP=list(kind=2, inorm=3)` and scaling the results by 2 (to convert double-sided spectra to single-sided spectra).

Let's assume `mtapspec` doesn't remove a mean and trend from the input series. We can do this easily with the `prewhiten` methods[3].

```
> require(RSEIS)
> dt=1 # km
> ats <- prewhiten(ts(magsat$clean, deltat=dt), plot=FALSE)
```

and if we set `AR.max` higher than zero, the program would've fit an auto-regressive (AR) model to the data. In Figure 3 we show the AR fitting method, and note that while we set `AR.max` relatively high, only and AR(6) model was fit significantly.

We didn't necessarily need to deal with the sampling information since it is just 1, but suppose the sampling information was based on an interval. We can account for an interval by using a negative value for `X.frq`, with which `psdcore` will interpret as an interval (instead of a frequency). A quick example highlights the equivalency:

```
> a <- rnorm(32)
> all.equal(psdcore(a,1)$spec, psdcore(a,-1)$spec)

[1] TRUE
```

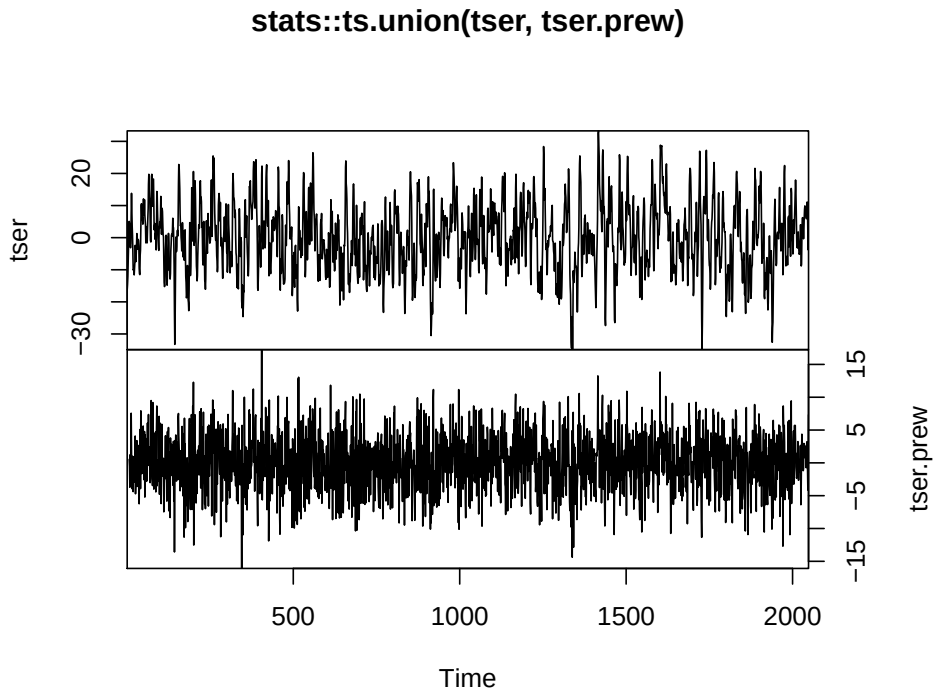
Returning the the `RSEIS` comparison, we first estimate the PSD from `mtapspec` with 10 tapers:

```
> tapinit <- 10
> Mspec <- mtapspec(ats, deltat(ats), MTP=list(kind=2, inorm=3, nwin=tapinit, npi=0))
> str(Mspec)

List of 12
 $ dat     : ts [1:2048, 1] -16.23 -14.56 -12.02 -7.21 -3.13 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : NULL
  ..- attr(*, "tsp")= num [1:3] 1 2048 1
 $ dt      : num 1
 $ spec    : num [1:4096] 528 557 600 595 615 ...
 $ dof     : num [1:4096] 20 20 20 20 20 20 20 20 20 20 ...
 $ Fv      : num [1:4096] 4.45e-20 4.78e-02 5.36e-01 1.54 1.15 ...
 $ Rspec   : num [1:2049, 1:10] 1.86e-07 -9.32e+01 6.05e+02 1.16e+03 -2.97e+02 ...
 $ Ispec   : num [1:2049, 1:10] 0 -227 -569 665 1157 ...
 $ freq    : num [1:2049] 0 0.000244 0.000488 0.000732 0.000977 ...
 $ df      : num 0.000244
 $ numfreqs: num 2049
 $ klen    : num 4096
 $ mtm     :List of 4
  ..$ kind : num 2
  ..$ nwin : num 10
  ..$ npi  : num 0
  ..$ inorm: num 3
```

[3] Although the name implies parametric fitting the default operation is to assume a linear model of the form $f(x) = \alpha x + \beta + \epsilon$.

```
> atsar<-prewhiten(ats, AR.max=100, verbose=FALSE)
```




```
> plot(psdcore(atsar,ntaper=10), log="dB", main="PSD of MAGSAT innovations")
```

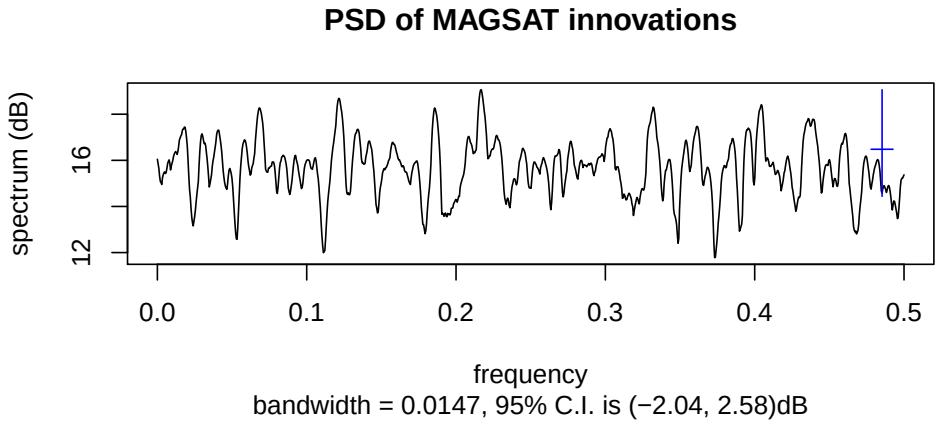


Figure 3: Top: Pre-whitening of a series assuming an AR model. Bottom: Power spectral density estimates of the innovations (model residuals): the spectrum is flat, as we would hope.

where `nwin` is the number of tapers taken and `npi` is, from the documentation, the "number of Pi-prolate functions" (we leave it out for the sake of comparison). Note that the object returned is not of class `spec`.

Then we calculate the comparative spectra from

1. `spectrum` (20% cosine taper),
2. `psdcore` (with fixed tapers), and
3. `pspectrum` (allowing adaptive taper refinement)

We will need to correct for normalization factors, as necessary.

```
> Xspec <- spec.pgram(ats, pad=1, taper=0.2, detr=TRUE, dem=TRUE, plot=FALSE)
> Pspec <- psdcore(ats, dt, tapinit)
> Aspec <- pspectrum(ats, dt, tapinit, niter=2)
> # Correct for double-sidedness of spectrum and mtapspec results
> nt <- 1:Mspec$numfreqs
> mspec <- Mspec$spec[nt] * 2
> Xspec$spec <- 2 * Xspec$spec
```

An easy comparison is to plot them on the same scale, as shown in Figure 4.

Because we did not specify the length of the FFT in `mtapspec` we end up with different length spectra. So, to form some statistical measure of the results, we need to interpolate PSD levels onto the `rlpSpec`-based frequencies:

```
> require(signal)
> pltpi <- interp1(pltf, pltp, Pspec$freq)
```

We then regress the spectral values from `mtapspec` against the `psdcore` results, since they both produced uniformly tapered spectra.

```
> df <- data.frame(x=dB(Pspec$spec), y=pltpi, tap=unclass(Aspec$taper))
> summary(dflm <- lm(y ~ x + 0, df))

Call:
lm(formula = y ~ x + 0, data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-3.0996 -0.3012  0.2784  0.8471  4.5689

Coefficients:
  Estimate Std. Error t value Pr(>|t|)
x 0.989507   0.001921     515   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.126 on 1024 degrees of freedom
Multiple R-squared: 0.9962,        Adjusted R-squared: 0.9962
F-statistic: 2.652e+05 on 1 and 1024 DF,  p-value: < 2.2e-16

> df$res <- residuals(dflm)
```

We create `ggplot2` objects for plotting purposes:

```
> require(RColorBrewer)
> cols <- c("dark grey", brewer.pal(8, "Set1")[c(5:4,2)])
> lwds <- c(1,2,2,5)
> plot(Xspec, log="dB", ylim=40*c(-0.4,1), ci.col=NA,
+        col=cols[1], lwd=lwds[1], main="PSD comparisons")
> pltf <- Mspec$freq
> lines(pltf, pltp <- dB(mspec), col=cols[2], lwd=lwds[2])
> plot(Pspec, log="dB",  add=TRUE, col=cols[3], lwd=lwds[3])
> plot(Aspec, log="dB", add=TRUE, col=cols[4], lwd=lwds[4])
> legend("topright",
+             c("spec.pgram","mtapspec","psdcore","pspectrum"),
+             title="Estimator", lwd=3, cex=1.1, col=cols)
```

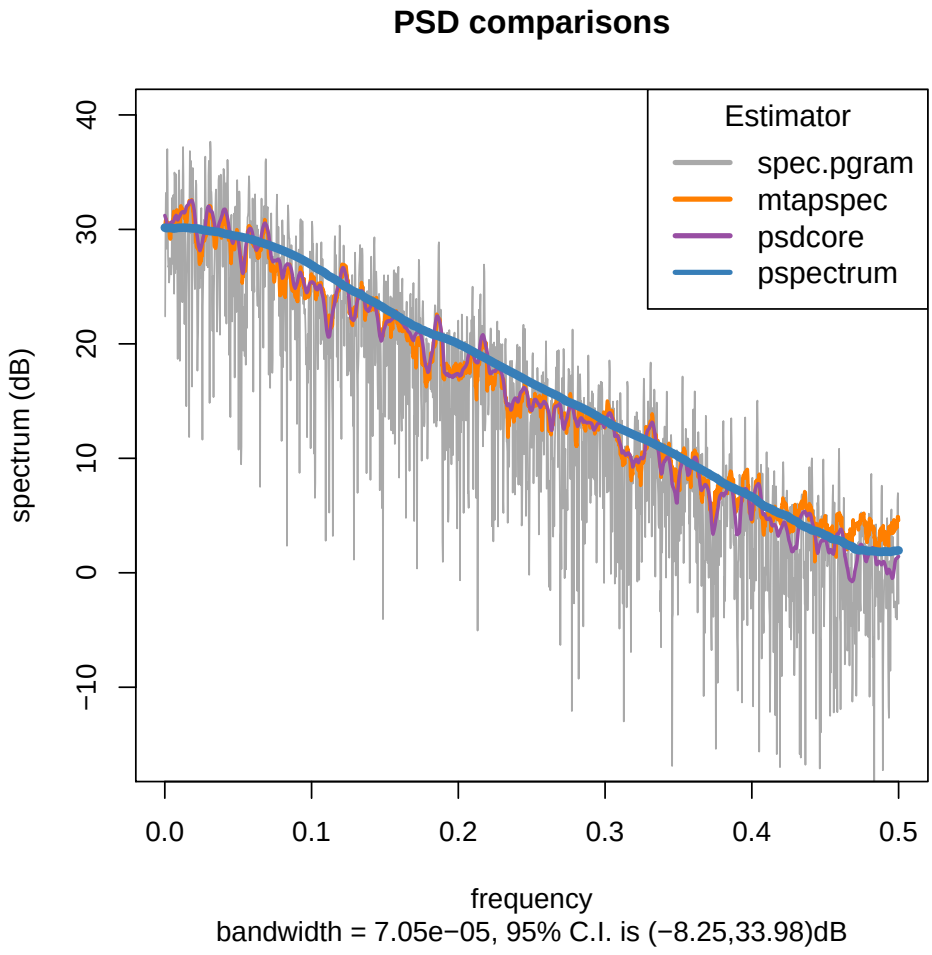


Figure 4: Comparisons of power spectral density estimators.

```
> require(ggplot2)
> g1 <- ggplot(df, aes(x=x, y=y)) +
+     geom_abline(intercept=0, slope=1, size=2, color="salmon")+
+     geom_point(aes(color=dB(y/x))) +
+     geom_smooth(colour="black", formula = y ~ x + 0, method="lm",
+                              se=TRUE, fullrange=TRUE) +
+     scale_colour_gradient2(mid="light grey") +
+     theme_bw() +
+     ggtitle("Regression of mtapspec against psdcore")
> g2 <- ggplot(df, aes(x=x, y=res)) +
+     geom_abline(intercept=0, slope=0, size=2, color="salmon") +
+     geom_point(aes(color=tap)) +
+     theme_bw() +
+     ggtitle("Regression residuals, colored by optimized tapers")
```

The regression and residuals from these objects are shown in Figure 5. The structure visible in the residuals at low power levels appears to be from curvature bias in the `mtapspec` results, which manifests at short wavelengths in Figure 4.

2.3 `multitaper::spec.mtm`

The function with the highest similarity to `rlpSpec` is `spec.mtm` in the `multitaper` package. This is not surprising since it calls source code of a Fortran equivalent to `rlpSpec` authored by R.L. Parker (2013). There are some notable differences, however. By default it uses the Discrete Prolate Spheroidal Sequences (dpss) of Thomson (1982), which demand bandwidth tuning there can be many more knobs to turn.

2.4 `SDF::sapa`

As of this writing, the package has no maintainer; lest we end up discussing deprecated and archived functions, we will not compare it to `rlpSpec`.

3 Assessing spectral properties

It is important to place bounds on the uncertainties associated with a spectral estimate. In a multitaper algorithm the uncertainty and resolution vary with the number of tapers in which the number of tapers controls the effective degrees of freedom ν of a χ^2_ν distribution. This goes approximately as

$$\sqrt{(10 * K/12)^{-1}}$$

and can be calculated with `spectral_properties`. For example:

```
> spp <- spectral_properties(Pspec$taper)
> spa <- spectral_properties(Aspec$taper)
> str(spa)

'data.frame':        1025 obs. of  5 variables:
 $ taper     : int  256 255 254 254 254 254 254 255 256 255 ...
 $ stderr    : num  0.0685 0.0686 0.0687 0.0687 0.0687 ...
 $ resolution: num  0.125 0.124 0.124 0.124 0.124 ...
 $ dof       : num  512 510 508 508 508 508 508 510 512 510 ...
 $ bw        : num  0.25 0.249 0.248 0.248 0.248 ...
```

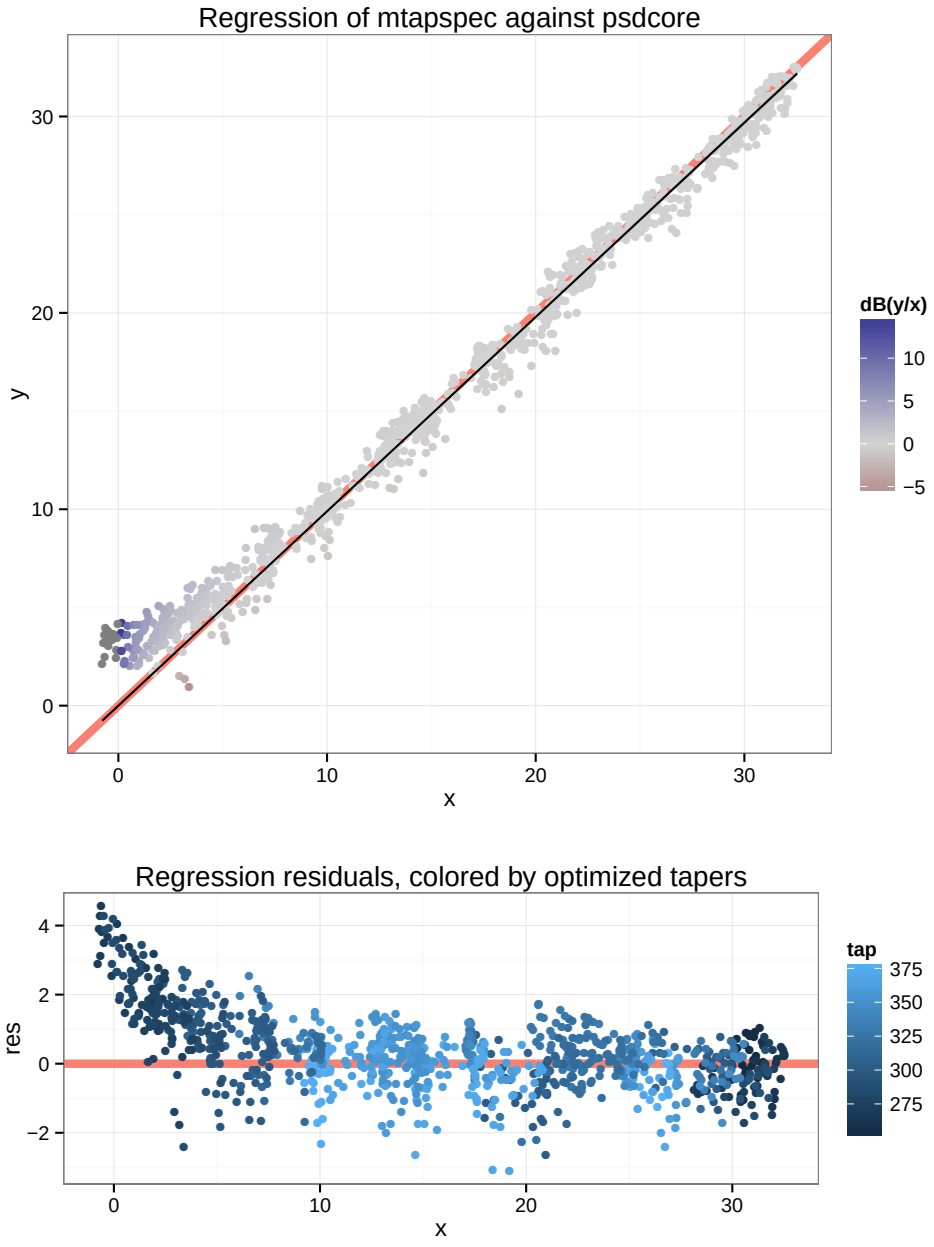
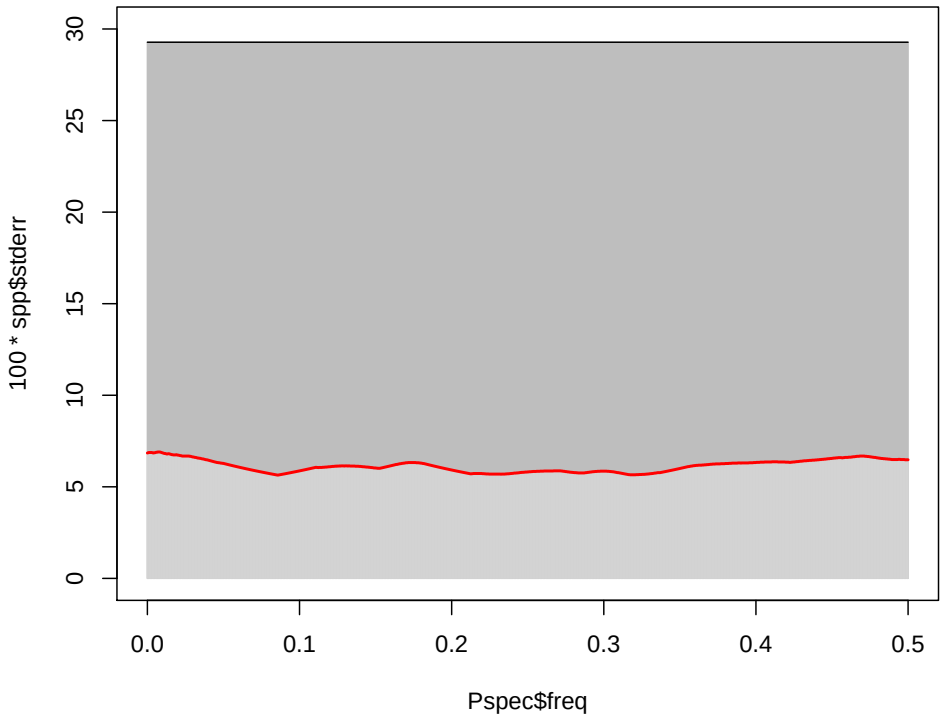


Figure 5: Regression of `mtapspec` PSD against `psdcore` PSD.

```
> create_poly <- function(x, y, dy){
+   xx <- c(x, rev(x))
+   yy <- c(y*(1+dy), rev(y*(1-dy)))
+   return(list(xx=xx, yy=yy))
+ }
> pspp <- create_poly(Pspec$freq, Pspec$spec, spp$stderr)
> pspa <- create_poly(Aspec$freq, Aspec$spec, spa$stderr)

> plot(Pspec$freq, 100*spp$stderr, type="h", col="grey", ylim=c(0,30))
> lines(Pspec$freq, 100*spp$stderr)
> lines(Aspec$freq, 100*spa$stderr, type="h", col="light grey")
> lines(Aspec$freq, 100*spa$stderr, col="red", lwd=2)
```



So uncertainty is reduced as the number of tapers is increased. Even though a multitaper spectral estimate can have much lower uncertainty than a naïve estimator, the amount of bias introduced by tapers may not be trivial. The approach of Riedel and Sidorenko (1995) is to find the number of tapers which minimizes any potential bias.

```
> plot(c(0,0.5),c(-8,35),col="white",
+        main="MAGSAT Spectral Uncertainty (p > 0.95)",
+        ylab="", xlab="spatial frequency, 1/km", yaxt="n", frame.plot=FALSE)
> lines(c(2,1,1,2)*0.01,c(5,5,8.01,8.01)-8)
> text(.05, -1.4, "3.01 dB")
> polygon(pspp$xx, dB(pspp$yy), col="light grey", border="black")
> text(0.15, 6, "With adaptive\ntaper refinement", cex=1.2)
> polygon(pspa$xx, dB(pspa$yy)-8, col="light grey", border="red")
> text(0.35, 22, "Uniform tapering", cex=1.2)
```

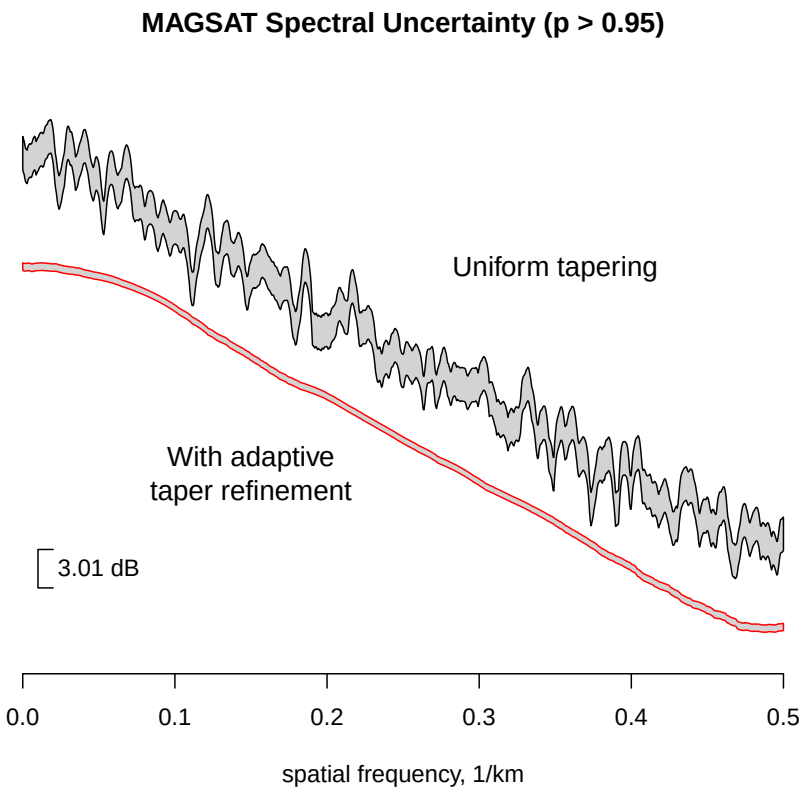


Figure 6: Spectral uncertainties with and without adaptive taper optimization.

4 Call overview

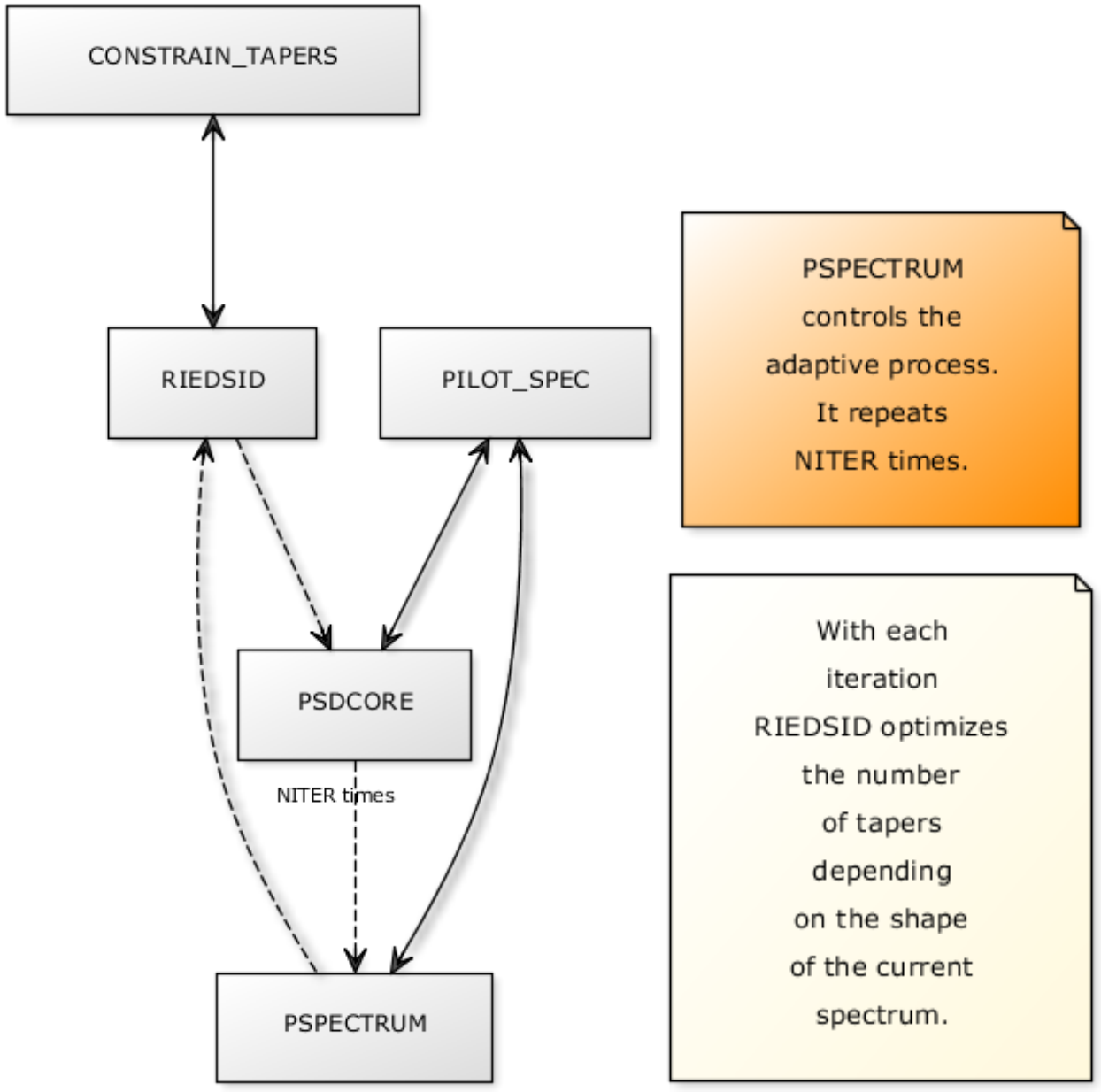


Figure 7: Simplified call graph for `rlpSpec`. The dashed lines show a simplified circuit in which the spectra and its tapers make during the iterative process.

References

Agnew, D. C. (1992). The time-domain behavior of power-law noises. *Geophysical Research Letters*, 19:333–336.

Langel, R., Ousley, G., Berbert, J., Murphy, J., and Settle, M. (1982). The MAGSAT mission. *Geophysical Research Letters*, 9(4):243–245.

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. (2013). PSD. `http://igppweb.ucsd.edu/%7Eparker/Software/#psd`. *Maintained software (last accessed 30 Jan 2013).*

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.

Riedel, K. S. and Sidorenko, A. (1995). Minimum bias multiple taper spectral estimation. *IEEE Trans. SP*, 43(1):188–195.

Thomson, D. J. (1982). Spectrum estimation and harmonic analysis. *Proceedings of the IEEE*, 70(9):1055–1096.