`rlpSpec`: Adaptive sine multitaper power spectral density estimation

Andrew J. Barbour and Robert L. Parker

February 12, 2013

Abstract

This vignette provides an overview of some of the features included in `rlpSpec`, designed to compute estimates of power spectral density (PSD) for a univariate series in a sophisticated manner, 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 based on the spectra from the previous iteration. Assuming the adaptive procedure converges, this produces power spectra with significantly lower spectral variance relative to results from less-sophisticated estimators. Sine tapers exhibit excellent leakage suppression characteristics, so bias effects are also reduced. Resolution and uncertainty vary with the number of tapers, which means we do not need to resort to either (1) windowing methods, which inherently degrade resolution at low-frequency (e.g. Welch's method); or (2) smoothing kernels, which can badly distort important features without careful tuning (e.g. the Daniell kernel in `stats::spectrum`). In this regards `rlpSpec` is best suited for data having large dynamic range and some mix of narrow and wide-band structure, features typically found in geophysical datasets.

Contents

1 Quick start: A minimal example.

First, we load the package into the namespace:

```
library(rlpSpec)

## Loading required package:  fftw
## Loaded rlpSpec (0.1.0) -- Adaptive multitaper spectrum estimation.
```

For a series to analyze, we can use `magnet`, included in `rlpSpec`, which represents along-track measurements of horizontal magnetic-field strength from a gimbaled, airborne magnetometer. These data are a small subset of the full Project MAGNET series (Coleman, 1992), which has provided insight into the history of the Earth's oceanic crust (Parker and O'Brien, 1997; O'Brien et al., 1999; Korte et al., 2002). The sampling interval is once every kilometer (km), so the will represent crustal magnetization with wavelengths longer than 2 km.

```
data(magnet)
```

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

```
names(magnet)

## [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(magnet, abs(mdiff) > 0)

##     km    raw  clean  mdiff
## 403  0  209.1 -3.635 -212.7
## 717  0 -248.7 -9.778  238.9
```

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(magnet$raw)

## Stage 0 est.  (pilot)
## detrending (and demeaning)
## Stage 1 est.  (Ave.  S.V.R. -4.1 dB)
## Stage 2 est.  (Ave.  S.V.R. -12.7 dB)
## Stage 3 est.  (Ave.  S.V.R. -18.6 dB)
## Stage 4 est.  (Ave.  S.V.R. -25.0 dB)
## Stage 5 est.  (Ave.  S.V.R. -35.1 dB)
## Normalized single-sided PSD (rlpspec) to sampling-freq.  1
```

```
psdc <- pspectrum(magnet$clean)

## Stage 0 est.  (pilot)
## detrending (and demeaning)
## Stage 1 est.  (Ave.  S.V.R. -3.9 dB)
## Stage 2 est.  (Ave.  S.V.R. -14.6 dB)
## Stage 3 est.  (Ave.  S.V.R. -22.0 dB)
## Stage 4 est.  (Ave.  S.V.R. -28.5 dB)
## Stage 5 est.  (Ave.  S.V.R. -34.4 dB)
## Normalized single-sided PSD (rlpspec) to sampling-freq.  1
```

Each `pspectrum` command calculates a pilot PSD, followed by `niter` iterations of refinement. With each iteration the number of tapers is adjusted based the proposed optimal number from Riedel and Sidorenko (1995), which depends on spectral shape; we use quadratically weighted spectral derivatives to estimate this shape. By default, a multipanel summary plot of the final PSD compared to the raw periodogram estimate is shown after the final iterative stage. Note that if the user forgets to assign the results of `pspectrum` to the global environment, this can be done with the `rlp_envGet` function:

```
psdc_recovered <- rlp_envGet("final_psd")
all.equal(psdc, psdc_recovered)

## [1] TRUE
```

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 1 km^{-1}, the assumed value.

Figure 1 compares the power spectra for the `raw` and `clean` series[2]. We expect the Project MAGNET data to be linear in the space of linear-frequencies and logarithmic-power; we see a clear improvement in spectral shape between the two series, simply because the large outliers have been removed. The PSD of the clean series shows a very "red" spectrum typical of geophysical processes (Agnew, 1992), and a rolloff in signal for 10 kilometer wavelengths and longer; whereas, the PSD for the raw series looks somewhat unrealistic at higher wavelengths– features which could be difficult to judge if the spectral variance was higher.

2 Comparisons with other methods

As we have shown in the Project MAGNET 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 compare results from `rlpSpec` with those from a few of the methods in Table 1, using the same data: the cleaned Project MAGNET series.

[1] Messages given by `pspectrum` with "Ave. S.V.R." (not shown) are in reference to the average spectral-variance reduction, found from double-differenced spectra at 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`.

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)

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 `psdcore` is an option to compare the results with a 20% tapered periodogram. The cosine estimator is found with the following command:

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

Within `psdcore` the comparison is made with the logical argument in `preproc` passed to `spec.pgram`, which is True by default.

As a matter of bookkeeping and good practice, we should consider the working environment accessed by `rlpSpec` functions. To ensure `psdcore` does not access any inappropriate information leftover from the previous calculations, we can set `refresh=TRUE`. We can then re-calculate the multitaper PSD and the raw periodogram with `plotpsd=TRUE`; these results are shown in Figure 2.

2.2 `RSEIS::mtapspec`

In `RSEIS` the spectrum estimation tool is `mtapspec`, which calls the program of Lees and Park (1995). There are numerous optional tuning parameters, including flags for normalization and taper averaging. For our purpose 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).

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

```
require(RSEIS)

## Loading required package:  RSEIS
## Loading required package:  RPMG
## Loading required package:  Rwave
##
## Attaching package:  'Rwave'
```

```
## The following object(s) are masked from 'package:stats':
##
## kernel

dt = 1  # km
# prewhiten the data after adding a linear trend + offset
summary(prewhiten(mc <- (ts(magnet$clean + 1000, deltat = dt) + seq_along(magnet$clean)),
    plot = FALSE))

## detrending (and demeaning)

##         Length Class  Mode
## lmdfit    12   lm     list
## ardfit     0   -none- NULL
## prew_lm 2048   ts     numeric
## prew_ar    0   -none- NULL
## imputed    1   -none- logical
```

Although the default operation of `prewhiten` is to fit a linear model of the form $f(x) = \alpha x + \beta + \epsilon$ using ordinary linear least squares, setting `AR.max` higher than zero to fit an auto-regressive (AR) model to the data[3]. This fit uses the Akaike infomation criterion (AIC) to select the highest order appropriate for the data.

```
summary(atsar <- prewhiten(mc, AR.max = 100, plot = FALSE))

## detrending (and demeaning)
## autoregressive model fit (returning innovations)

##         Length Class  Mode
## lmdfit    12   lm     list
## ardfit    14   ar     list
## prew_lm 2048   ts     numeric
## prew_ar 2048   ts     numeric
## imputed    1   -none- logical

print(atsar$ardfit)

##
## Call:
## ar.yw.default(x = tser_prew_lm, aic = TRUE, order.max = AR.max,     demean = TRUE)
##
## Coefficients:
##      1       2       3       4       5       6
##  1.513  -1.104   0.672  -0.388   0.211  -0.079
##
## Order selected 6  sigma^2 estimated as  19.5
```

[3]Note that the linear trend fitting is removed from the series prior to AR estimation, and the residuals from this fit are also returned.

```
ats_lm <- atsar$prew_lm
ats_ar <- atsar$prew_ar
```

We didn't necessarily need to deal with the sampling information since it is just 1 per km; but, supposing the sampling information was based on an interval, we could have used a negative value for `X.frq`, with which `psdcore` would 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_lm, deltat(ats_lm), MTP = list(kind = 2, inorm = 3, nwin = tapinit,
    npi = 0))
```

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`:

```
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
```

We will calculate the comparative spectra from

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

and we will need to correct for normalization factors, as necessary, with `normalize`. Note that by default the normalization is set within `pspectrum` (with `normalize`) once the adaptive procedure is finished.

```
Xspec <- spec.pgram(ats_lm, pad = 1, taper = 0.2, detr = TRUE, dem = TRUE, plot = FALSE)
Pspec <- psdcore(ats_lm, dt, tapinit)
Aspec <- pspectrum(ats_lm, dt, tapinit, plot = FALSE)

## Stage 0 est.  (pilot)
## detrending (and demeaning)
## Stage 1 est.  (Ave.  S.V.R. -1.5 dB)
## Stage 2 est.  (Ave.  S.V.R. -8.8 dB)
## Stage 3 est.  (Ave.  S.V.R. -16.3 dB)
## Stage 4 est.  (Ave.  S.V.R. -18.1 dB)
## Stage 5 est.  (Ave.  S.V.R. -23.6 dB)
## Normalized single-sided PSD (rlpspec) to sampling-freq.  1

# Correct for double-sidedness of spectrum and mtapspec results
class(Mspec)

## [1] "list"

Mspec <- normalize(Mspec, dt, "spectrum")

## Normalized double-sided PSD (spectrum) to sampling-freq.  1

nt <- 1:Mspec$numfreqs
mspec <- Mspec$spec[nt]
class(Xspec)

## [1] "spec"

Xspec <- normalize(Xspec, dt, "spectrum")

## Normalized double-sided PSD (spectrum) to sampling-freq.  1
```

These estimates are shown on the same scale 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 can interpolate PSD levels onto the `rlpSpec`-based frequencies (or reciprocally):

```
require(signal)

## Loading required package:  signal
## Loading required package:  MASS
```

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

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

We regress the spectral values from `mtapspec` against the `psdcore` results because we have used them to produce uniformly tapered spectra with an equal number of sine tapers.

```
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
## -2.586 -0.327  0.210  0.933  5.382
##
## Coefficients:
##   Estimate Std. Error t value Pr(>|t|)
## x  0.99217    0.00202     492   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.18 on 1024 degrees of freedom
## Multiple R-squared: 0.996,Adjusted R-squared: 0.996
## F-statistic: 2.42e+05 on 1 and 1024 DF,  p-value: <2e-16
```

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

Create `ggplot2` objects for plotting purposes:

```
require(ggplot2)
```

```
## Loading required package:  ggplot2
```

```
# data and regression
g <- ggplot(df, aes(x = x, y = y))
g1 <- g + 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)
# and the residuals
g <- ggplot(df, aes(x = x, y = res))
g2 <- g + geom_abline(intercept = 0, slope = 0, size = 2, color = "salmon") +
    geom_point(aes(color = tap))
```

We show this regression and its associated residuals in Figure 5. The structure visible in the residuals at low power levels may 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: it uses the sine multitapers, and can adaptively refine the spectrum. In fact, this function calls source code of a Fortran equivalent to `rlpSpec` authored by R.L. Parker (2013) to do these operations.

There are some notable differences, though. By default `spec.mtm` uses the Discrete Prolate Spheroidal Sequences (dpss) of Thomson (1982), which can have very good spectral leakage suppression (assuming the number of tapers used is appropriate for the desired resolution, which varies inversely with the time-bandwidth product). Spectral analyses using dpss can have superior results if the series is relatively short (e.g. $N < 1000$), or has inherent spectra with sharply changing features or deep wells. Improper usage of the dpss, however, can lead to severe bias. Thus, considerable care should be given to parameter choices, which translates practicably to having many more knobs to turn.

2.4 `sapa::SDF`

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 Can AR prewhitening improve the spectrum?

This question must be addressed on a case-by-base basis; but, if there is significant auto-regressive structure in the series then the answer is likely Yes. The MAGNET dataset is an example where removing the spectrum of the innovations from the AR prewhitening procedure can reduce spectral variance considerably.

Recall the results of the prewhitening in Section 2.2. While `AR.max` was set relatively high, only an AR(6) model was fit significantly, according to the AIC requirements. The estimated variance of the innovations is about 20 nT^2. If the innovation spectrum is flat (as we expect), this variance translates to power levels of about 16 decibels for a 1 km sampling interval.

```
ntap <- 7
psd_ar <- psdcore(ats_ar, ntaper = ntap, refresh = TRUE)
dB(mean(psd_ar$spec))

## [1] 15.82
```

An example of this potential variance-reduction is shown in Figure 6, which we have used `pilot_spec` to remove the AR spectrum internally. The AR component adds approximately ± 3 dB to the original spectrum, and removing it will give a considerably smoother spectrum without the need for adaptive taper optimization.

4 Assessing spectral properties

4.1 Spectral uncertainties

It is important to place bounds on the uncertainties associated with a spectral estimate. In a multitaper algorithm the uncertainty is distributed as a χ^2_ν variate where ν is the number of degrees of freedom, which

is twice the number of tapers applied. A proxy for this is simply $1/\sqrt{\nu - 1}$. Using $\nu = 2 * K$ we can approximate the distribution of uncertainties from the tapers alone; however, a more rigorous estimate comes from evaluating the appropriate distribution for a coverage probability (e.g. $p = 0.95$). Among other calculations, `spectral_properties` returns the χ^2_ν based confidence intervals for $p = 0.95$, as well as the approximate uncertainties.

To illustrate, we plot the uncertainties for an integer sequence[4] of tapers $[0, 50]$, shown in Figure 7. The benefits of having more than just a few tapers becomes obvious, though the spectral uncertainty is asymptotically decreasing with taper numbers and yields only slight improvements with logarithmic number of tapers.

Returning to the Project MAGNET spectra, let us compare the `rlpSpec` spectra with spectral uncertainty as bounded polygons. First calculate the uncertainty polygon data:

```
spp <- spectral_properties(Pspec$taper, db.ci = TRUE)
spa <- spectral_properties(Aspec$taper, db.ci = TRUE)
str(spa)

## 'data.frame': 1025 obs. of  8 variables:
##  $ taper            : int  235 235 235 235 235 235 235 235 235 235 ...
##  $ stderr.chi.lower : num  -0.54 -0.54 -0.54 -0.54 -0.54 ...
##  $ stderr.chi.upper : num  0.572 0.572 0.572 0.572 0.572 ...
##  $ stderr.chi.median: num  0.195 0.195 0.195 0.195 0.195 ...
##  $ stderr.chi.approx: num  0.196 0.196 0.196 0.196 0.196 ...
##  $ resolution       : num  0.46 0.46 0.46 0.46 0.46 ...
##  $ dof              : num  470 470 470 470 470 470 470 470 470 470 ...
##  $ bw               : num  0.23 0.23 0.23 0.23 0.23 ...

create_poly <- function(x, y, dy) {
    xx <- c(x, rev(x))
    yy <- c(y + dy, rev(y - dy))
    return(data.frame(xx = xx, yy = yy))
}
pspp <- create_poly(Pspec$freq, dB(Pspec$spec), spp$stderr.chi.approx)
psppu <- create_poly(Pspec$freq, dB(Pspec$spec), spp$stderr.chi.upper)
pspa <- create_poly(Aspec$freq, dB(Aspec$spec), spa$stderr.chi.approx)
pspau <- create_poly(Aspec$freq, dB(Aspec$spec), spa$stderr.chi.upper)
```

and plot the comparison, shown in Figure 8.

4.2 Spectral resolution

There is an inherent tradeoff between the number of tapers applied and the spectral resolution (effectively, the spectral bandwidth). In general, the greater the number of tapers applied, the lower the spectral resolution. We can use the information returned from `spectral_properties` to visualize the actual differences in resolution for the Project MAGNET PSD estimates; these are shown in Figure 9.

[4] Note the χ^2_ν distribution is defined for non-negative, non-integer degrees of freedom, but we cannot apply fractions of tapers.

4.3 Visualizing the adaptive history

One might be curious to study how the uncertainties change with each iteration. `pspectrum` saves an array of "historical" data in its working environment. Specifically, it saves the frequencies, spectral values, and number of tapers at each stage of the adaptive procedure, accessible with `get_adapt_history`. To ensure a fresh calculation and to add a few more iterations to visualize, we repeat the adaptive spectral analysis, and then bring the stage history into the `.GlobalEnv` environment:

```
pspectrum(ats_lm, niter = 6, plot = FALSE)

## Stage 0 est.  (pilot)
## detrending (and demeaning)
## Stage 1 est.  (Ave.  S.V.R. -3.9 dB)
## Stage 2 est.  (Ave.  S.V.R. -14.6 dB)
## Stage 3 est.  (Ave.  S.V.R. -22.0 dB)
## Stage 4 est.  (Ave.  S.V.R. -28.5 dB)
## Stage 5 est.  (Ave.  S.V.R. -34.4 dB)
## Stage 6 est.  (Ave.  S.V.R. -35.7 dB)
## Normalized single-sided PSD (rlpspec) to sampling-freq.  1

str(AH <- get_adapt_history())

## List of 3
##  $ freq    : num [1:1025] 0 0.000488 0.000977 0.001465 0.001953 ...
##  $ stg_kopt:List of 7
##   ..$ :Class 'tapers'  int [1:1025] 7 7 7 7 7 7 7 7 7 ...
##   ..$ :Class 'tapers'  int [1:1025] 22 22 23 24 25 24 23 22 21 22 ...
##   ..$ :Class 'tapers'  int [1:1025] 79 78 77 76 77 78 77 76 75 76 ...
##   ..$ :Class 'tapers'  int [1:1025] 202 201 200 199 198 197 196 195 194 193 ...
##   ..$ :Class 'tapers'  int [1:1025] 223 223 223 223 223 223 223 223 222 222 ...
##   ..$ :Class 'tapers'  int [1:1025] 264 264 264 264 264 264 264 264 265 265 ...
##   ..$ :Class 'tapers'  int [1:1025] 306 306 306 306 306 306 306 306 306 306 ...
##  $ stg_psd :List of 7
##   ..$ : num [1:1025] 1144 1228 1318 1363 1377 ...
##   ..$ : num [1:1025] 1085 1121 1158 1225 1331 ...
##   ..$ : num [1:1025] 1526 1523 1521 1502 1499 ...
##   ..$ : num [1:1025] 1559 1555 1551 1549 1548 ...
##   ..$ : num [1:1025] 1068 1069 1071 1072 1072 ...
##   ..$ : num [1:1025] 969 970 970 969 973 ...
##   ..$ : num [1:1025] 974 975 976 977 979 ...
```

Followed by some trivial manipulation:

```
Freqs <- (AH$freq)
Dat <- AH$stg_psd
numd <- length(Freqs)
numit <- length(Dat)
StgPsd <- dB(matrix(unlist(Dat), ncol = numit))
```

```
Dat <- AH$stg_kopt
StgTap <- matrix(unlist(Dat), ncol = numit)
rm(Dat, AH)
```

We can plot these easily with `matplot` or other tools. We show the adaptive history in Figure 10. It may be informative to investigate cross correlation coefficients between the stages:

```
symnum(cT <- cor(StgTap))

## Warning:  the standard deviation is zero

##
## [1,] 1
## [2,] ? 1
## [3,] ? . 1
## [4,] ?   , 1
## [5,] ?   . . 1
## [6,] ?       . 1
## [7,] ?     .   + 1
## attr(,"legend")
## [1] 0 ' ' 0.3 '.' 0.6 ',' 0.8 '+' 0.9 '*' 0.95 'B' 1 \t    ## NA: '?'
```

```
symnum(cP <- cor(StgPsd))

##
## [1,] 1
## [2,] B 1
## [3,] B B 1
## [4,] B B B 1
## [5,] B B B B 1
## [6,] B B B B B 1
## [7,] B B B B B B 1
## attr(,"legend")
## [1] 0 ' ' 0.3 '.' 0.6 ',' 0.8 '+' 0.9 '*' 0.95 'B' 1
```

But, in this case, only the PSD estimates are significantly correlated.

5 Call overview

Shown in Figure 11 is a flow chart highlighting the essential functions involved in the adaptive estimation process. The primary function is `pspectrum`.

6 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] ggplot2_0.9.3      signal_0.7-2       MASS_7.3-22
##  [4] RColorBrewer_1.0-5 RSEIS_3.0-9        Rwave_2.0
##  [7] RPMG_2.1-4         rlpSpec_0.1-0      fftw_1.0-3
## [10] knitr_0.9
##
## loaded via a namespace (and not attached):
##  [1] Peaks_0.2        colorspace_1.2-0 dichromat_1.2-4  digest_0.6.0
##  [5] evaluate_0.4.3   formatR_0.7      gtable_0.1.2     labeling_0.1
##  [9] lattice_0.20-10  munsell_0.4      plyr_1.8         proto_0.3-10
## [13] reshape2_1.2.2   scales_0.2.3     stringr_0.6.2    zoo_1.7-9
```

References

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

Coleman, R. J. (1992). Project Magnet high-level vector survey data reduction. In *Types and Characteristics of Data for Geomagnetic Field Modeling*, volume 3153, pages 215–248.

Korte, M., Constable, C., and Parker, R. (2002). Revised magnetic power spectrum of the oceanic crust. *Journal of Geophysical Research*, 107(B9):2205.

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

O'Brien, M. S., Parker, R. L., and Constable, C. G. (1999). Magnetic power spectrum of the ocean crust on large scales. *Journal of Geophysical Research*, 104(B12):29189–29.

Parker, R. L. (2013). PSD. `http://igppweb.ucsd.edu/%7Eparker/Software/`. *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.

Parker, R. L. and O'Brien, M. S. (1997). Spectral analysis of vector magnetic field profiles. *Journal of Geophysical Research*, 102(B11):24815–24.

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.

```
plot(psdc, log = "dB", main = "Raw and Clean Project MAGNET power spectral density",
    lwd = 3, ci.col = NA, ylim = c(0, 32), yaxs = "i")
# plot(psdc_ar, log='dB', add=TRUE, lwd=3, col='red')
plot(psdr, log = "dB", add = TRUE, lwd = 3, lty = 5)
text(c(0.25, 0.34), c(11, 24), c("Clean", "Raw"), cex = 1)
```

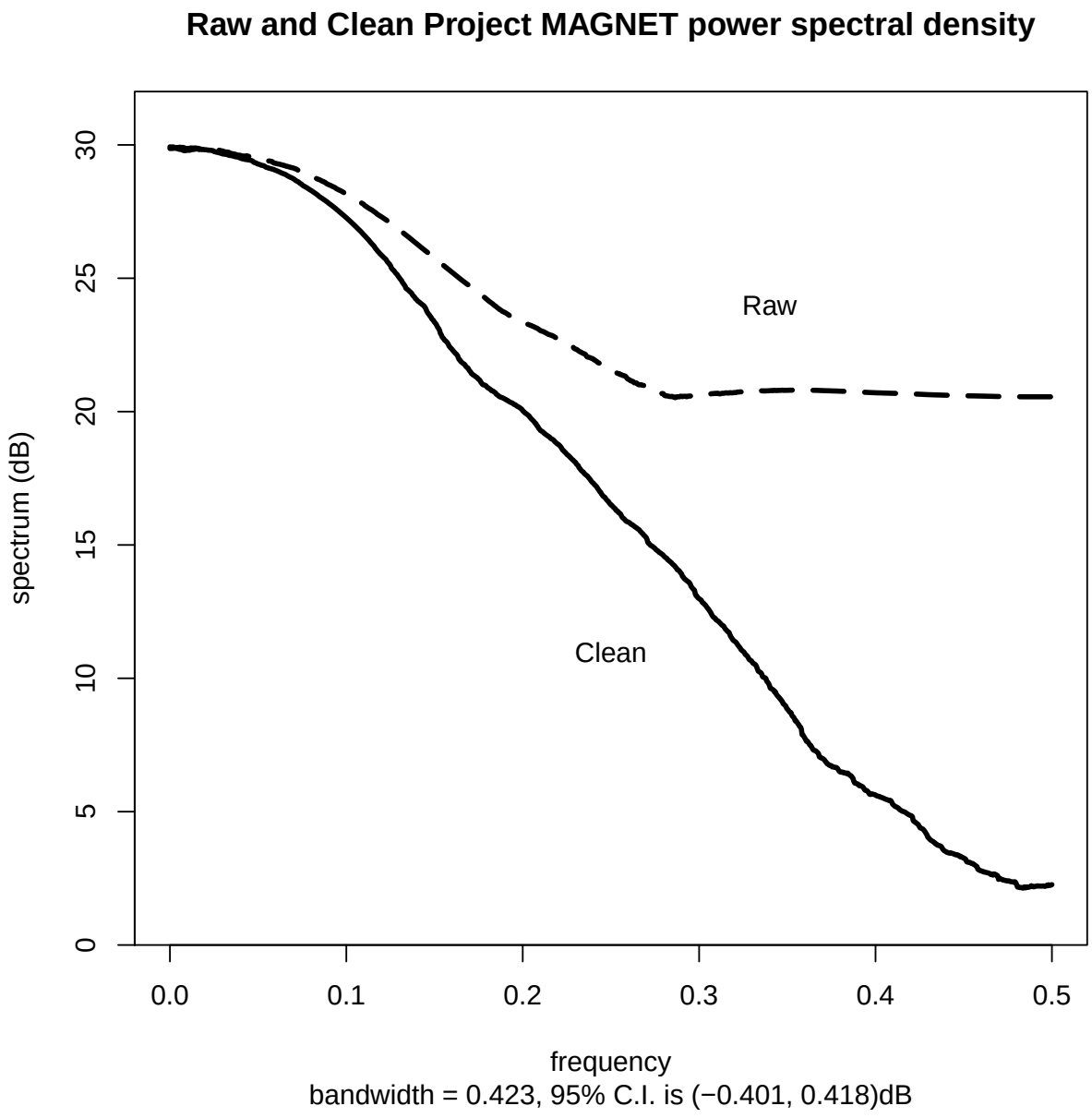

Figure 1: Power spectral density estimates for the raw and cleaned `Project MAGNET` data bundled with `rlpSpec`. Note that because the class is `'spec'` we have utilized existing methods in the `stats` namespace.

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

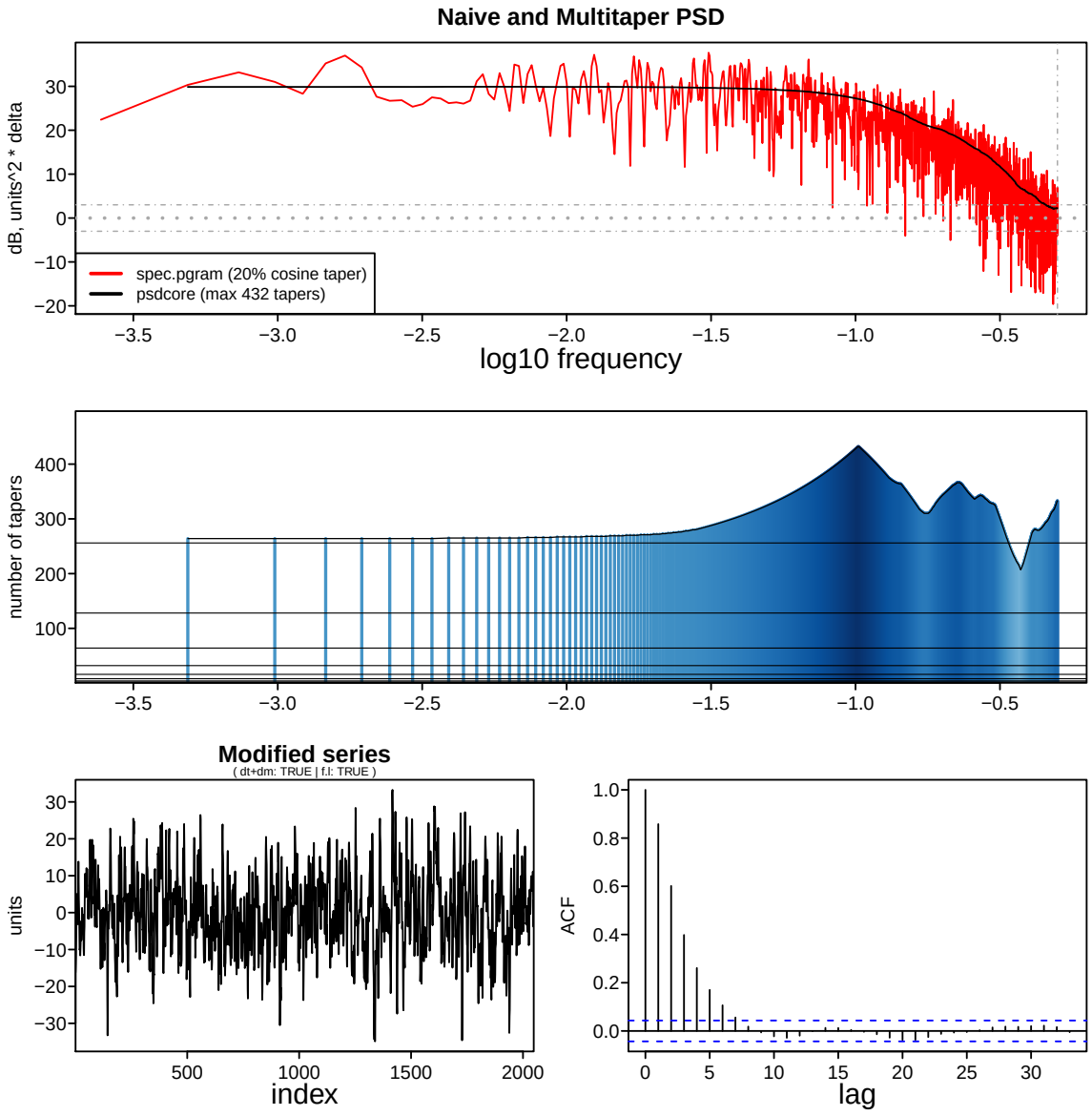

Figure 2: Top: Comparison between PSD estimators for the clean Project MAGNET data. The frequency axis is in units of $\log_{10}$ km^{-1}, and power axis is in decibels. Middle: The number of tapers applied as a function of frequency from the `plot.tapers` method. Bottom: The spatial series used to estimate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

```
plot(ts.union(orig.plus.trend = mc, linear = ats_lm, ar = ats_ar), yax.flip = TRUE,
    main = sprintf("Prewhitened Project MAGNET series"))
mtext(sprintf("linear and linear+AR(%s)", atsar$ardfit$order), line = 1.1)
```

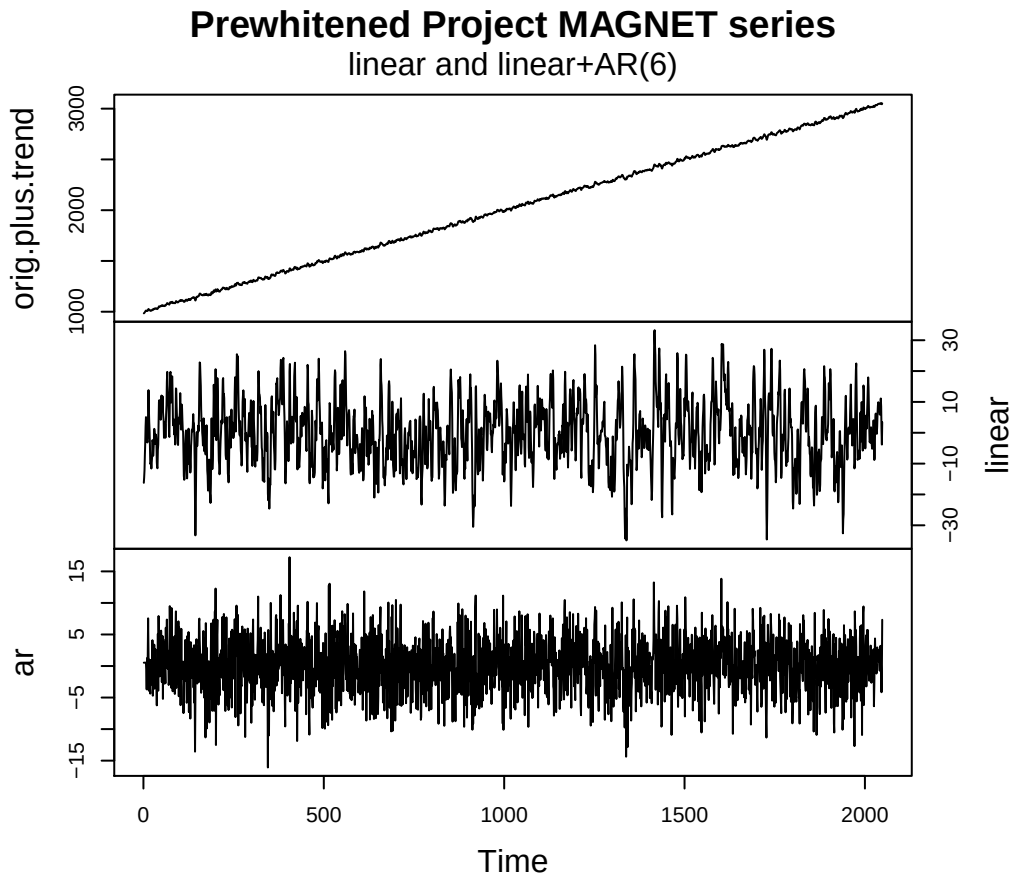

Figure 3: Pre-whitening of the Project MAGNET series (with a synthetic linear model superimposed on it) assuming linear and linear-with-AR models.

```
require(RColorBrewer)
                        ## Loading required package:  RColorBrewer

cols <- c("dark grey", brewer.pal(8, "Set1")[c(5:4, 2)])
lwds <- c(1, 2, 2, 5)
par(las = 1)
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", "RSEIS::mtapspec", "psdcore", "pspectrum"),
    title = "Estimator", lwd = 3, cex = 1.1, col = cols)
```

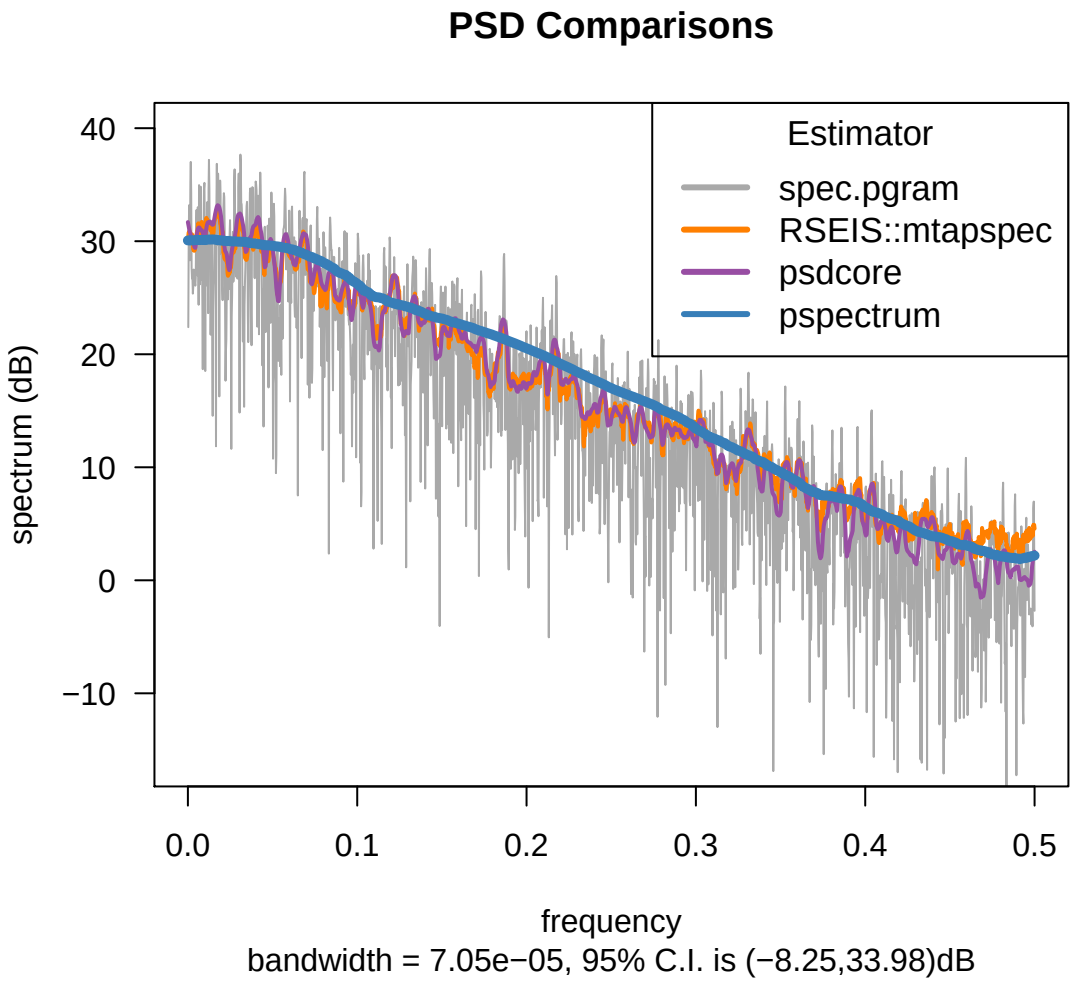

Figure 4: Comparisons of estimations of Project MAGNET power spectral densities.

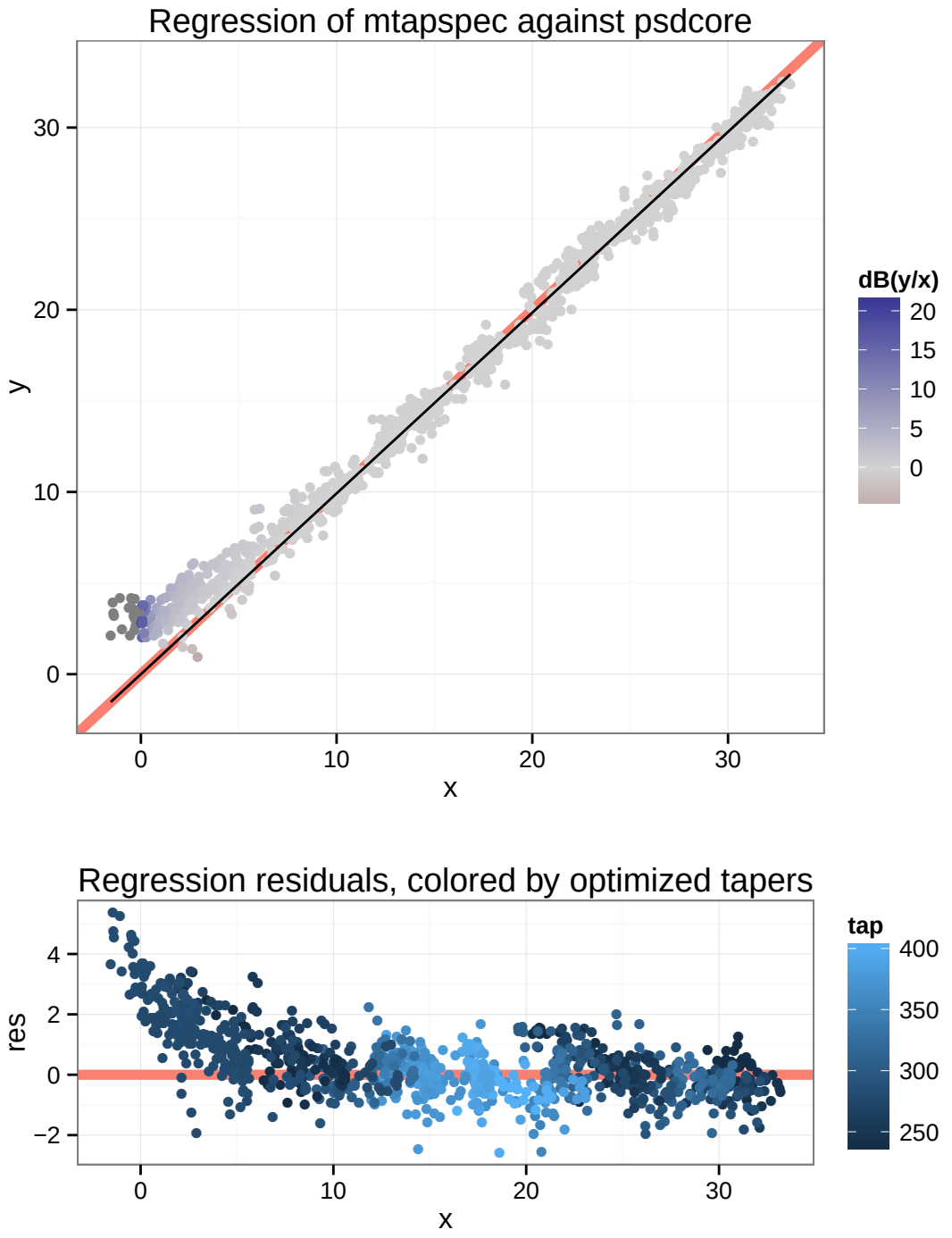

Figure 5: Regression of Project MAGNET PSD estimates: `mtapspec` against `psdcore`.

```
pilot_spec(ats_lm, ntap = ntap, remove.AR = 100, plot = TRUE)
plot(Aspec, log = "dB", add = TRUE, col = "grey", lwd = 4)
plot(Aspec, log = "dB", add = TRUE, lwd = 3, lty = 3)
```

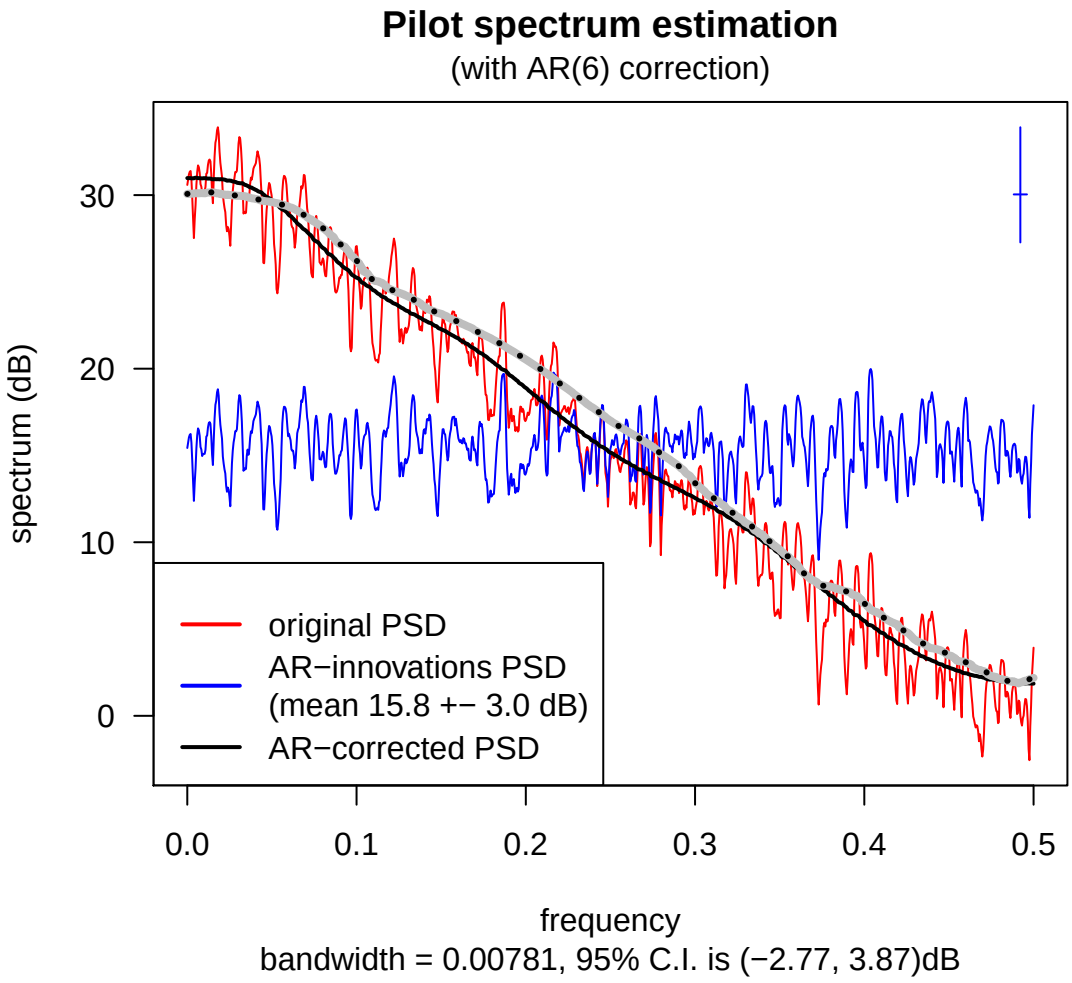

Figure 6: Spectral variance may be reduced by removing the effect on an AR process, as is shown for the MAGNET data using `pilot_spec`. The dotted line is the adaptively estimated spectrum from Figure 4.

```
sp <- spectral_properties(as.tapers(1:50), p = 0.95, db.ci = TRUE)
par(las = 1)
plot(stderr.chi.upper ~ taper, sp, type = "s", ylim = c(-10, 20), yaxs = "i",
    xaxs = "i", xlab = expression("number of tapers (" * nu/2 * ")"), ylab = "dB",
    main = "Spectral uncertainties")
mtext("(additive factor)", line = 0.3)
lines(stderr.chi.lower ~ taper, sp, type = "s")
lines(stderr.chi.median ~ taper, sp, type = "s", lwd = 2)
lines(stderr.chi.approx ~ taper, sp, type = "s", col = "red", lwd = 2)
# to reach 3 db width confidence interval at p=.95
abline(v = 33, lty = 3)
legend("topright", c(expression("Based on " * chi^2 * "(p," * nu * ") and (1-p," *
    nu * ")"), expression("" * chi^2 * "(p=0.5," * nu * ")"), "approximation"),
    lwd = c(1, 3, 3), col = c("black", "black", "red"), bg = "white")
```

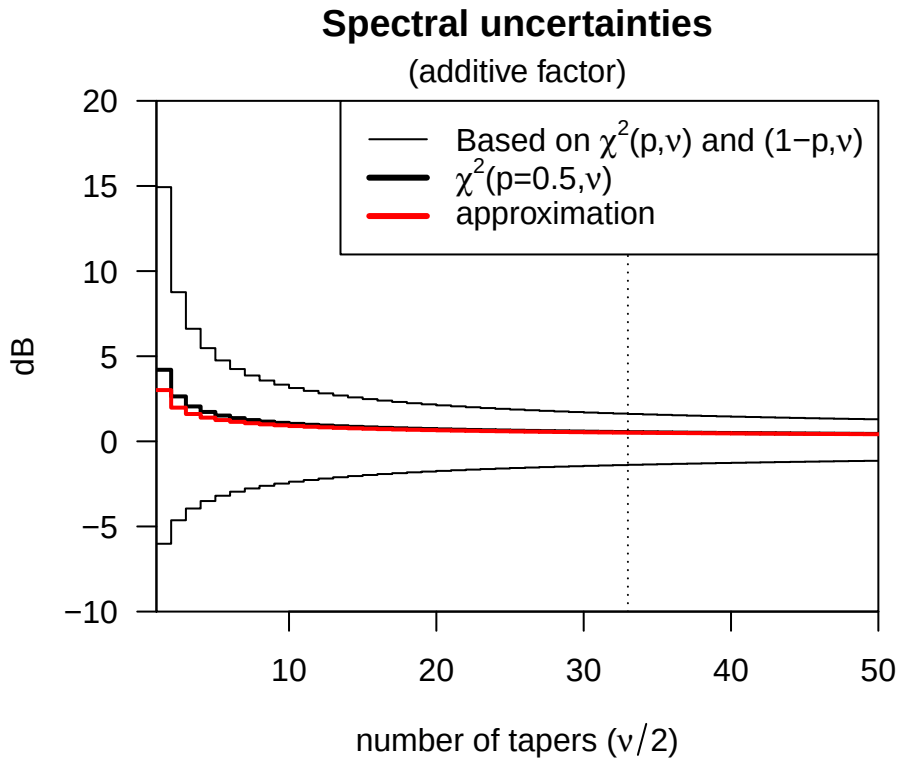

Figure 7: Additive spectral uncertainties by number of tapers. These quantized curves are found by evaluating the χ^2_ν distribution, where ν is the number of degrees of freedom (two per taper). The black lines show uncertainties for a coverage probability of 0.95. The thick, red line shows an approximation to these uncertainties based on $1/\sqrt{\nu - 1}$, which is accurate to within a few percent in most cases.

```
plot(c(0, 0.5), c(-8, 35), col = "white", main = "Project MAGNET 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(0.05, -1.4, "3.01 dB")
polygon(psppu$xx, (psppu$yy), col = "light grey", border = "black", lwd = 0.5)
polygon(pspp$xx, (pspp$yy), col = "dark grey", border = NA)
text(0.15, 6, "With adaptive\ntaper refinement", cex = 1.2)
polygon(pspau$xx, (pspau$yy) - 10, col = "light grey", border = "black", lwd = 0.5)
polygon(pspa$xx, (pspa$yy) - 10, col = "dark grey", border = NA)
text(0.35, 22, "Uniform tapering", cex = 1.2)
```

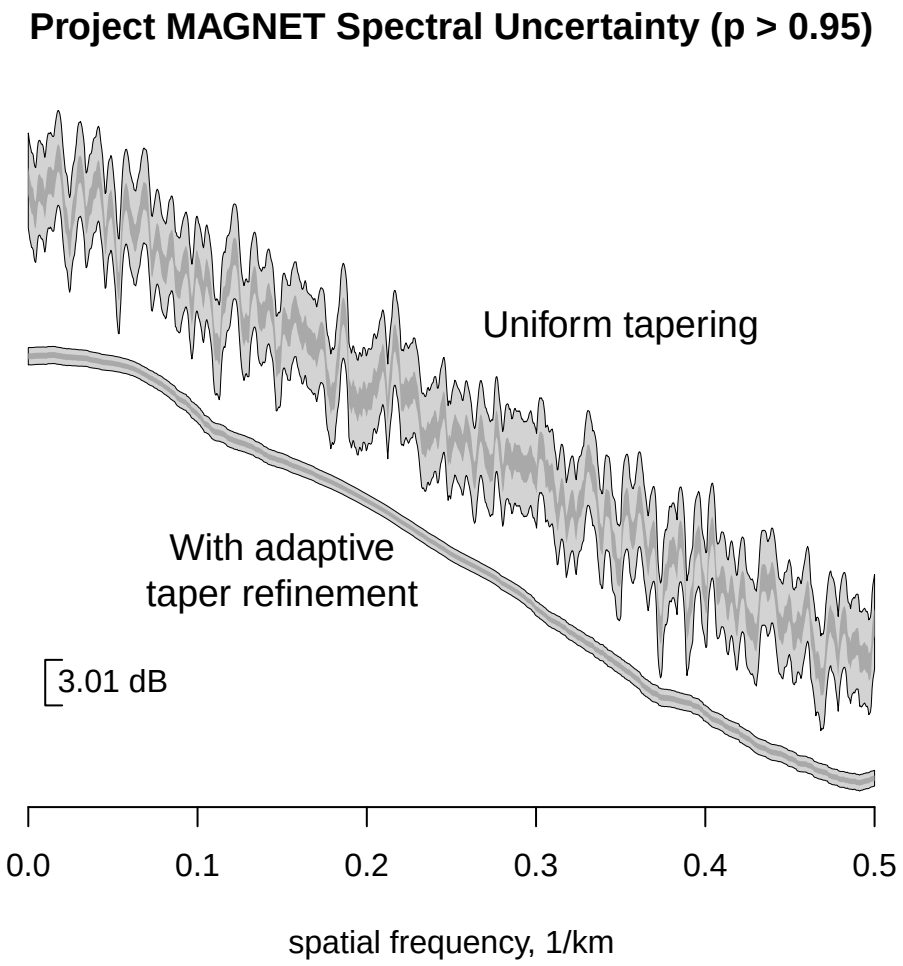

Figure 8: Project MAGNET Spectral uncertainties with and without adaptive taper optimization. The filled regions encompass the upper limit and approximate values of spectral uncertainty .

```
frq <- Aspec$freq
relp <- dB(1/spa$resolution)
par(las = 1)
plot(frq, relp, col = "light grey", ylim = dB(c(1, 5)), type = "h", xaxs = "i",
    yaxs = "i", ylab = "dB", xlab = "frequency, 1/km", main = "Project MAGNET Spectral Resolution and Un
lines(frq, relp)
lines(frq, spp$stderr.chi.upper + relp, lwd = 1.5, lty = 3)
lines(frq, spa$stderr.chi.upper + relp, lwd = 3, lty = 2)
abline(h = dB(sqrt(vardiff(Aspec$spec))), lwd = 1.5, lty = 2, col = "red")
```

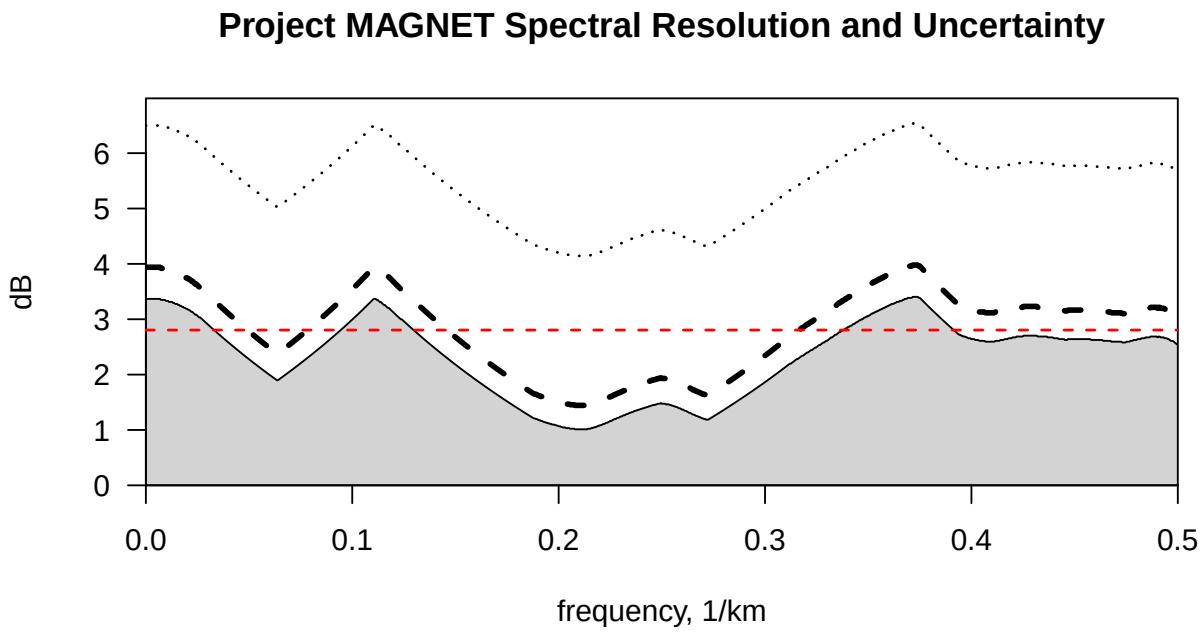

Figure 9: Resolution limits for Project MAGNET PSD (`niter=5` adapts–the default) in decibels relative to the spectrum. The thick dashed line shows the resolved uncertainties ($p = 0.05$) in the adaptively refined spectrum; whereas, the thin dashed line shows the approximate RMS spectral variation. The dotted line shows the resolved uncertainties for the uniformly tapered result.

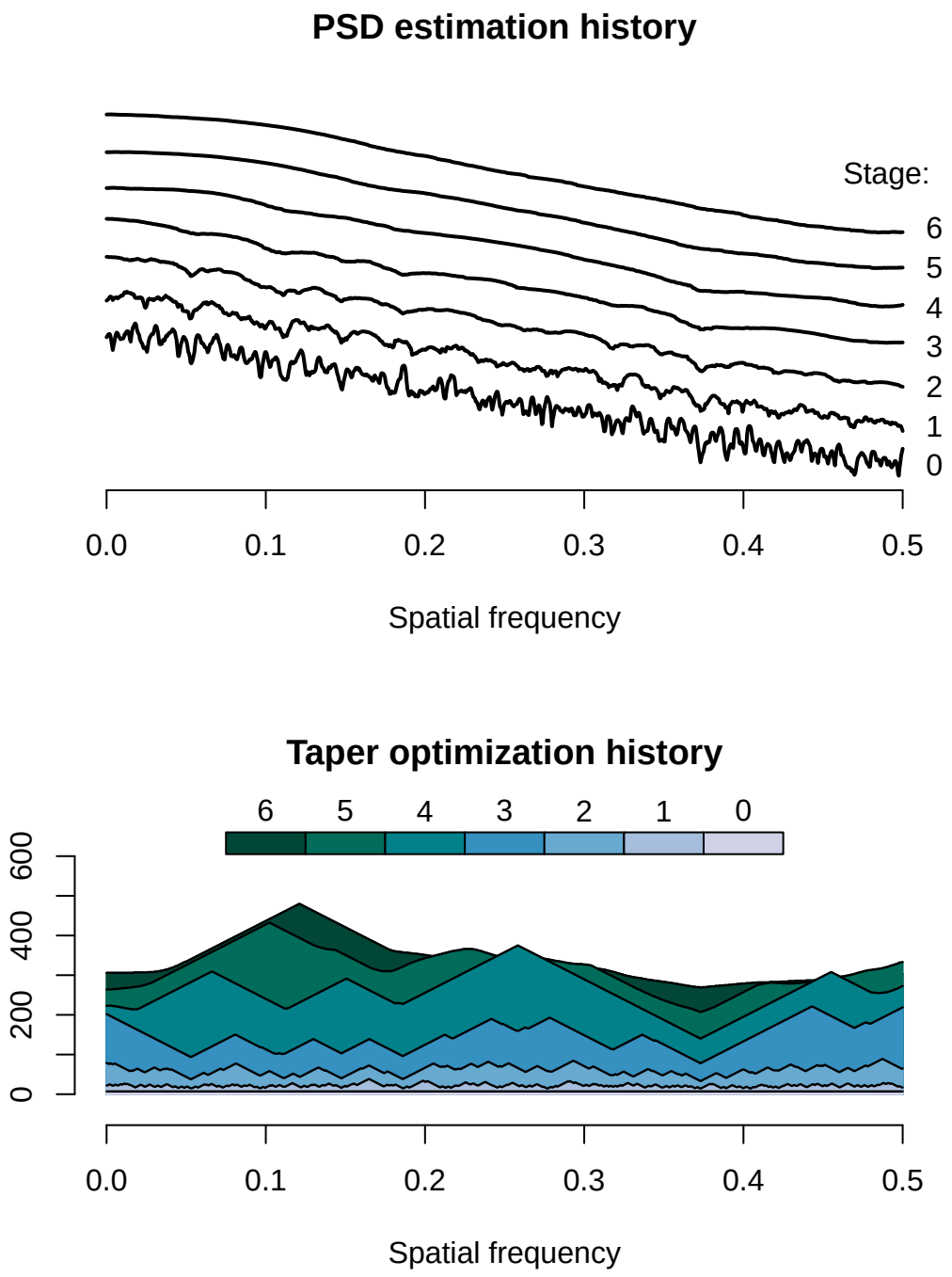

Figure 10: Adaptive spectral estimation history. Top: Sequential PSD series for each stage of the adaptive method, offset by a few decibels for visualization purposes. Bottom: Filled polygons showing the number of tapers at each stage.

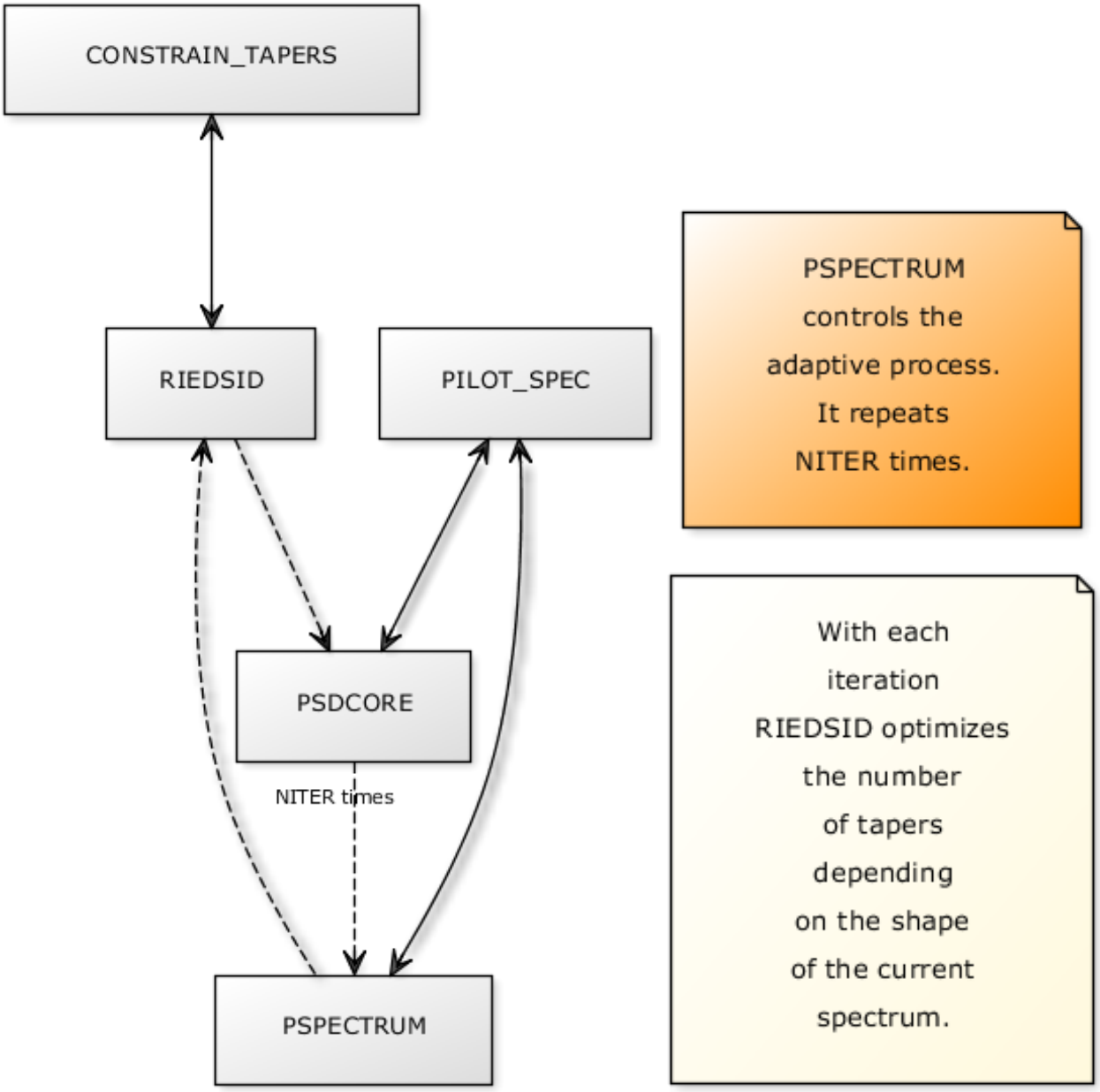

Figure 11: Simplified call graph for `rlpSpec`. The dashed lines show a simplified circuit in which the spectra and its tapers make during the iterative process.