

# Package ‘sapa’

January 2, 2012

**Type** Package

**Title** Spectral Analysis for Physical Applications

**Version** 1.1-0

**Date** 2011-10-016

**Author** William Constantine and Donald Percival (Applied Physics Laboratory, University of Washington)

**Maintainer** William Constantine <wlbconstan@gmail.com>

**Depends** R (>= 2.5.1), ifulertools (>= 1.1-0), splus2R (>= 1.1-0)

**Description** Software for the book Spectral Analysis for Physical Applications, Donald B. Percival and Andrew T. Walden, Cambridge University Press, 1993.

**License** GPL-2

**Collate** sapa\_sdf.R sapa\_taper.R sapa\_pkg.R

**Repository** CRAN

**Date/Publication** 2011-10-17 18:42:39

## R topics documented:

ACVS . . . . .	2
SDF . . . . .	3
taper . . . . .	7

<b>Index</b>	<b>11</b>
--------------	-----------

---

ACVS	<i>Autocovariance sequence</i>
------	--------------------------------

---

### Description

Calculates the autocovariance sequence for an input time series.

### Usage

```
ACVS(x, biased=TRUE, center=TRUE)
```

### Arguments

x	a numeric vector representing a uniformly sampled real-valued time series.
biased	a logical value. If TRUE, the biased estimator (normalized by $N$ , the number of samples in the time series) is returned. If FALSE, the result is the unbiased estimator (the $k$ th ACVS value is normalized by $N -  k $ for the unbiased case where $k = 0, \dots, N - 1$ ). Default: TRUE.
center	a logical value. If TRUE, the series is first centered (sample mean is subtracted from series) prior to calculating the ACVS. Default: TRUE.

### Value

a numeric vector containing the single-sided ACVS for lags  $k = 0, \dots, N - 1$  where  $N$  is the length of the input time series.

### See Also

[SDF](#).

### Examples

```
## calculate the ACVS for an N(0,1) realization
plot(seq(0,99), ACVS(rnorm(100)), type="l", lwd=2,
      xlab="lag", ylab="ACVS(rnorm(100))")
gridOverlay()
```

**Description**

Estimate the process (cross) spectral density function via nonparametric models.

**Usage**

```
SDF(x, method="direct", taper.=NULL, window=NULL,
    n.taper=5, overlap=0.5, blocksize=NULL,
    single.sided=TRUE, sampling.interval=NULL,
    center=TRUE, recenter=FALSE, npad=2*numRows(x))
```

**Arguments**

x	a vector or matrix containing uniformly-sampled real-valued time series. If a matrix, each column should contain a different time series.
blocksize	an integer representing the number of points (width) of each block in the WOSA estimator scheme. Default: $\text{floor}(N/4)$ where $N$ is the number of samples in each series.
center	a logical value. If TRUE, the mean of each time series is recentered prior to estimating the SDF. Default: TRUE.
method	a character string denoting the method to use in estimating the SDF. Choices are "direct", "lag window", "wosa" (Welch's Overlapped Segment Averaging), "multitaper". See <b>DETAILS</b> for more information. Default: "direct".
n.taper	an integer defining the number of tapers to use in a multitaper scheme. This value is overwritten if the taper input is of class taper. Default: 5.
npad	an integer representing the total length of each time series to analyze after padding with zeros. This argument allows the user to control the spectral resolution of the SDF estimates: the normalized frequency interval is $\Delta f = 1/\text{npad}$ . This argument must be set such that $\text{npad} > 2$ . Default: $2 \times \text{numRows}(x)$ .
overlap	a numeric value on $[0, 1]$ denoting the fraction of window overlap for the WOSA estimator. Default: 0.5.
recenter	a logical value. If TRUE, the mean of each time series is recentered after (possibly) tapering the series prior to estimating the SDF. Default: FALSE.
sampling.interval	a numeric value representing the interval between samples in the input time series $x$ . Default: NULL, which serves as a flag to obtain the sampling interval via the <code>deltat</code> function. If $x$ is a list, the default sampling interval is <code>deltat(x[[1]])</code> . If $x$ is an atomic vector (ala <code>isVectorAtomic</code> ), then the default samplign interval is established ala <code>deltat(x)</code> . Finally, if the input series is a matrix, the sampling interval of the first series (assumed to be in the first column) is obtained ala <code>deltat(x[, 1])</code> .

single.sided	a logical value. If TRUE, a single-sided SDF estimate is returned corresponding to the normalized frequency range of $[0, 1/2]$ . Otherwise, a double-sided SDF estimate corresponding to the normalized frequency interval $[-1/2, 1/2]$ is returned. Default: TRUE.
taper.	an object of class <code>taper</code> or a character string denoting the primary taper. If an object of class <code>taper</code> , the length of the taper is checked to ensure compatibility with the input <code>x</code> . See <b>DETAILS</b> for more information. The default values are a function of the method as follows: <b>direct</b> normalized rectangular taper <b>lag window</b> normalized Parzen window with a cutoff at $N/2$ where $N$ is the length of the time series. <b>wosa</b> normalized Hanning taper <b>multitaper</b> normalized Hanning taper
window	an object of class <code>taper</code> or a character string denoting the (secondary) window for the lag window estimator. If an object of class <code>taper</code> , the length of the taper is checked to ensure compatibility with the input <code>x</code> . See <b>DETAILS</b> for more information. Default: Normalized Hanning window.

## Details

Let  $X_t$  be a uniformly sampled real-valued time series of length  $N$ , Let an estimate of the process spectral density function be denoted as  $\hat{S}_X(f)$  where  $f$  are frequencies on the interval  $[-1/(2\Delta t), 1/(2\Delta t)]$  where  $\Delta t$  is the sampling interval. The supported SDF estimators are:

**direct** The direct SDF estimator is defined as  $\hat{S}_X^{(d)}(f) = |\sum_{t=0}^{N-1} h_t X_t e^{-i2\pi ft}|^2$ , where  $\{h_t\}$  is a data taper normalized such that  $\sum_{t=0}^{N-1} h_t^2 = 1$ . If  $h_t = 1/\sqrt{N}$  then we obtain the definition of the periodogram  $\hat{S}_X^{(p)}(f) = \frac{1}{N} |\sum_{t=0}^{N-1} X_t e^{-i2\pi ft}|^2$ . See the taper function for more details on supported window types.

**lag window** The lag window SDF estimator is defined as  $\hat{S}_X^{(lw)}(f) = \sum_{\tau=-(N-1)}^{N-1} w_\tau \hat{S}_{X,\tau}^{(d)} e^{-i2\pi f\tau}$ , where  $\hat{S}_{X,\tau}^{(d)}$  is the autocovariance sequence estimator corresponding to some direct spectral estimator (often the periodogram) and  $w_\tau$  is a lag window (popular choices are the Parzen, Papoulis, and Daniell windows). See the taper function for more details.

**wosa** Welch's Overlapped Segment Averaging SDF estimator is defined as

$$\hat{S}^{(wosa)} = \frac{1}{N_B} \sum_{j=0}^{N_B-1} \hat{S}_{jN_O}^{(d)}(f)$$

where

$$\hat{S}_l^{(d)}(f) \equiv \left| \sum_{t=0}^{N_S-1} h_t X_{t+l} e^{-i2\pi ft} \right|^2, \quad 0 \leq l \leq N - N_S;$$

Here,  $N_O$  is a positive integer that controls how much overlap there is between segments and that must satisfy both  $N_O \leq N_S$  and  $N_O(N_B - 1) = N - N_S$ , while  $\{h_t\}$  is a data taper appropriate for a series of length  $N_S$  (i.e.,  $\sum_{t=0}^{N_S-1} h_t^2 = 1$ ).

**multitaper** A multitaper spectral estimator is given by

$$\hat{S}_X^{(mt)}(f) = \frac{1}{K} \sum_{k=0}^{K-1} \left| \sum_{t=0}^{N-1} h_{k,t} X_t e^{-i2\pi ft} \right|^2,$$

where  $S(k, f) = \left| \sum_{t=0}^{N-1} h_{k,t} X_t \exp(-i2\pi ft) \right|^2$  and  $\{h_{k,t}\}$ ,  $k = 0, \dots, K-1$ , is a set of  $K$  orthonormal data tapers.

$$\sum_{t=0}^{N-1} h_{k,t} h_{k',t} = \begin{cases} 1, & \text{if } k = k'; \\ 0, & \text{otherwise} \end{cases}$$

Popular choices for multitapers include sinusoidal tapers and discrete prolate spheroidal sequences (DPSS). See the taper function for more details.

**Cross spectral density function estimation:** If the input  $x$  is a matrix, where each column contains a different time series, then the results are returned in a matrix whose columns correspond to all possible unique combinations of cross-SDF estimates. For example, if  $x$  has three columns, then the output will be a matrix whose columns are  $\{S_{11}, S_{12}, S_{13}, S_{22}, S_{23}, S_{33}\}$  where  $S_{ij}$  is the cross-SDF estimate of the  $i$ th and  $j$ th column of  $x$ . All cross-spectral density function estimates are returned as complex-valued series to maintain the phase relationships between components. For all  $S_{ij}$  where  $i = j$ , however, the imaginary portions will be zero (up to a numerical noise limit).

## Value

an object of class SDF.

## S3 METHODS

**as.matrix** converts the (cross-)SDF estimate(s) as a matrix. Optional arguments are passed directly to the `matrix` function during the conversion.

**plot** plots the (cross-)SDF estimate(s). Optional arguments are:

**xscale** a character string defining the scaling to perform on the (common) frequency vector of the SDF estimates. See the `scaleData` function for supported choices. Default: "linear".

**yscale** a character string defining the scaling to perform on the SDF estimates. See the `scaleData` function for supported choices. Default: "linear".

**type** a single character defining the plot type (ala the `par` function) of the SDF plots. Default: `ifelse(numRows(x) > 100, "l", "h")`.

**xlab** a character string representing the x-axis label. Default: "FREQUENCY (Hz)".

**ylab** a (vector of) character string(s), one per (cross-)SDF estimate, representing the y-axis label(s). Default: in the multivariate case, the strings " $S_{ij}$ " are used for the y-axis labels, where  $i$  and  $j$  are the indices of the different variables. For example, if the user supplies a 2-column matrix for  $x$ , the labels "S11", "S12", and "S22" are used to label the y-axes of the corresponding (cross-)SDF plots. In the univariate case, the default string "SDF" prepended with a string describing the type of SDF performed (such as "Multitaper") is used to label the y-axis.

**plot.mean** a logical value. If TRUE, the SDF value at normalized frequency  $f = 0$  is plotted for each SDF. This frequency is associated with the sample mean of the corresponding time series. A relatively large mean value dominates the spectral patterns in a plot and thus the corresponding frequency is typically not plotted. Default: !attr(x, "center").

**n.plot** an integer defining the maximum number of SDF plots to place onto a single graph. Default: 3.

**FUN** a post processing function to apply to the SDF values prior to plotting. Supported functions are Mod, Im, Re and Arg. See each of these functions for details. If the SDF is purely real (no cross-SDF is calculated), this argument is coerced to the Mod function. Default: Mod.

**add** A logical value. If TRUE, the plot is added using the current par() layout. Otherwise a new plot is produced. Default: FALSE.

... additional plot parameters passed directly to the genPlot function used to plot the SDF estimates.

**print** prints the object. Available options are:

**justify** text justification ala prettPrintList. Default: "left".

**sep** header separator ala prettyPrintList. Default: ":".

... Additional print arguments sent directly to the prettyPrintList function.

## References

Percival, Donald B. and Constantine, William L. B. (2005) "Exact Simulation of Gaussian Time Series from Nonparametric Spectral Estimates with Application to Bootstrapping", *Journal of Computational and Graphical Statistics*, accepted for publication.

D.B. Percival and A. Walden (1993), *Spectral Analysis for Physical Applications: Multitaper and Conventional Univariate Techniques*, Cambridge University Press, Cambridge, UK.

## See Also

[taper](#), [ACVS](#).

## Examples

```
## calculate various SDF estimates for the
## sunspots series. remove mean component for a
## better comparison.
data <- as.numeric(sunspots)
methods <- c("direct", "wosa", "multitaper",
            "lag window")

S <- lapply(methods, function(x, data) SDF(data, method=x), data)
x <- attr(S[[1]], "frequency")[-1]
y <- lapply(S, function(x) decibel(as.vector(x)[-1]))
names(y) <- methods

## create a stack plot of the data
stackPlot(x, y, col=1:4)

## calculate the cross-spectrum of the same
```

```

## series: all spectra should be the same in
## this case
SDF(cbind(data,data), method="lag")

## calculate the SDF using npad=31
SDF(data, npad=31, method="multitaper")

```

---

taper

*Oracle function for obtaining a particular taper/window*


---

## Description

Develop signal processing tapers or windows.

## Usage

```

taper(type="rectangle", n.sample=100, n.taper=NULL,
      sigma=0.3, beta=4*pi*(n.sample-1)/n.sample, cutoff=floor(n.sample/2),
      sidelobedB=80, roughness=n.sample/2, flatness=0.3,
      bandwidth=4, normalize=TRUE)

```

## Arguments

bandwidth	bandwidth for DPSS tapers. See Details for more information. Default: 4.
beta	kaiser window shape factor (must be positive or zero). See Details for more information. Default: $4\pi \cdot (n.sample - 1) / n.sample$ .
cutoff	parzen or Papoulis window cutoff (must be greater than unity). See Details for more information. Default: $\text{floor}(n.sample/2)$ .
flatness	raised cosine taper flatness fraction (must be on $[0,1]$ ). See Details for more information. Default: 0.3.
n.sample	an integer denoting the number of samples. Default: 1000.
n.taper	an integer defining the multitaper order (number of orthogonal tapers) to use in a multitaper scheme. The taper order directly impacts the quality of the SDF estimate. Low taper orders are usually associated with SDF estimates with low bias and high variance, while high taper orders attenuate the variance of the estimate at the risk of incurring a large bias. This tradeoff between bias and variance is unavoidable but taper order allows you to tune the SDF to meet the needs of your application. Studies show that a multitaper order of 5 typically provides a good balance with reasonably low bias and variance properties (see the references for more details). Default: NULL, which serves as a flag to set the default taper order depending on the type of taper chosen for the analysis. If sine or dpss multitapers are chosen, the default taper order is 5, otherwise is set to unity.
normalize	a logical value. If TRUE, the taper is normalized to have unit energy. Default: TRUE.

roughness	daniell window roughness factor (must be positive). See Details for more information. Default: <code>n.sample/2</code> .
sidelobeB	chebyshev sidelobeB bandwidth in decibels (must be positive). See Details for more information. Default: 80.
sigma	standard deviation for Gaussian taper. Default: 0.3.
type	a character string denoting the type of taper to create. Supported types are "rectangle", "triangle", "raised cosine", "hanning", "hamming", "blackman", "nuttall", "gaussian", "kaiser", "chebyshev", "born_jordan", "sine", "parzen", "papoulis", "daniell", and "dpss". See Details for more information. Default: "rectangle".

### Details

Let  $w(\cdot)$  and  $h(t)$  for  $t = 0, \dots, N-1$  be a *lag window* and *taper*, respectively. The following lag window or taper types are supported.

**rectangular** A rectangular taper is defined as  $h_t = 1$ .

**triangle** A triangular taper is defined as  $h_t = h_{N-t-1} = 2(t+1)/(N+1)$  for  $t < M$  where  $M = \lfloor N/2 \rfloor$  and  $h_M = 1$  if  $N$  is evenly divisible by 2.

**raised cosine** A raised cosine is a symmetric taper with a flat mid-plateau. Let  $p \in [0, 1]$  be the fraction of the length of the taper that is flat,  $M = \lfloor pN \rfloor$ , and  $\beta = 2\pi/(M+1)$ . A raised cosine taper is defined as

$$h_t = h_{N-t-1} = 0.5(1 - \cos(\beta(t+1))) \text{ for } 0 \leq t < \lfloor M/2 \rfloor$$

$$h_t = 1 \text{ for } \lfloor M/2 \rfloor \leq t < N - \lfloor M/2 \rfloor.$$

**hanning** Let  $\beta = 2\pi/(N+1)$ . A Hanning taper is defined as  $h_t = 0.5(1 - \cos(\beta(t+1)))$ .

**hamming** Let  $\beta = 2\pi/(N-1)$ . A Hamming taper is defined as  $h_t = 0.54 - 0.46 \cos(\beta t)$ .

**blackman** Let  $\beta = 2\pi/(N+1)$ . A Blackman taper is defined as  $h_t = 0.42 - 0.5 \cos(\beta(t+1)) + 0.08 \cos(2\beta(t+1))$ .

**nuttall** Let  $\beta = 2\pi/(N-1)$ . A Nuttall taper is defined as  $h_t = 0.3635819 - 0.4891775 \cos(\beta t) + 0.1365995 \cos(2\beta t) - 0.0106411 \cos(3\beta t)$

**gaussian** Let  $\sigma$  be the standard deviation of a Gaussian distribution. Let  $\beta(t) = 2\sigma(0.5 - t/(N-1))$ . A Gaussian taper is defined as

$$h_t = h_{N-t-1} = e^{-\beta^2/2} \text{ for } 0 \leq t < \lfloor N/2 \rfloor$$

$$h_{N/2} = 1 \text{ if } N \text{ is evenly divisible by } 2$$

**kaiser** Let  $V_t = (2t-1-N)/N$  and  $I_0(\cdot)$  be the zeroth-order modified Bessel function of the first kind. Given the shape factor  $\beta > 0$ , a Kaiser taper is defined as  $h_t = I_0(\beta\sqrt{1-V_t^2})/I_0(\beta)$ .

**chebyshev** The Dolph-Chebyshev taper is a function of both the desired length  $N$  and the desired sidelobe level (our routine accepts a sidelobe attenuation factor expressed in decibels). See the Mitra reference for more details.

**born jordan** Let  $M = (N - 1)/2$ . A Born-Jordan taper is defined as  $h_t = h_{N-t-1} = 1/(M - t + 1)$ .

**sine** Sine multitapers are defined as

$$h_{k,t} = \left( \frac{2}{N+1} \right)^{1/2} \sin \left( \frac{(k+1)\pi(t+1)}{N+1} \right),$$

for  $t = 0, \dots, N-1$  and  $k = 0, \dots, \dots$ . This simple equation defines a good approximation to the discrete prolate spheroidal sequences (DPSS) used in multitaper SDF estimation schemes.

**parzen** A Parzen lag window is defined as

$$w_{\tau,m} = \begin{cases} 1 - 6(t/m)^2 + 6(|t|/m)^3, & |t| \leq m/2; \\ 2(1 - |t|/m)^3, & m/2 < |t| \leq m/2; \\ 0, & \text{otherwise.} \end{cases}$$

for  $-(N-1) \leq \tau \leq (N-1)$ . The variable  $m$  is referred to as the *cutoff* since all values beyond that point are zero.

**papoulis** A Papoulis lag window is defined as

$$w_{\tau,m} = \begin{cases} \frac{1}{\pi} |\sin(\pi\tau/m)| + (1 - |\tau|/m) \cos(\pi\tau/m), & |\tau| < m; \\ 0, & |\tau| \geq m \end{cases}$$

for  $-(N-1) \leq \tau \leq (N-1)$ . The variable  $m$  is referred to as the *cutoff* since all values beyond that point are zero.

**daniell** A Daniell lag window is defined as

$$w_{\tau,m} = \begin{cases} \frac{\sin(\pi\tau/m)}{\pi\tau/m}, & |\tau| < N; \\ 0, & |\tau| \geq N \end{cases}$$

for  $-(N-1) \leq \tau \leq (N-1)$ . The variable  $m$  is referred to as the *roughness* factor, since, in the context of spectral density function (SDF) estimation, it controls the degree of averaging that is performed on the preliminary direct SDF estimate. The smaller the roughness, the greater the amount of smoothing.

**dpss** Discrete prolate spheroidal sequences are (typically) used for multitaper spectral density function estimation. The first order DPSS can be defined (to a good approximation) as

$$h_{t,0} = C \times I_0(\widetilde{W} \sqrt{1 - (1 - g_t)^2}) / I_0(\widetilde{W})$$

for  $t = 1, \dots, N$ , where  $C$  is a scaling constant used to force the normalization  $\sum h_{t,k}^2 = 1$ ;  $\widetilde{W} = \pi W(N-1)\Delta t$  where  $\Delta t$  is the sampling interval;  $g_t = (2t-1)/N$ ; and  $I_0(\cdot)$  is the modified Bessel function of the first kind and zeroth order. The parameter  $W$  is related to the *resolution bandwidth* since it roughly defines the desired half-width of the central lobe of the resulting spectral window. Higher order DPSS tapers (i.e.,  $h_{t,k}$  for  $k > 0$ ) can be calculated using a relatively simple tridiagonalization formulation (see the references for more information). Finally, we note that the sampling interval  $\Delta t$  can be set to unity without any loss of generality.

## Value

an object of class taper.

### S3 METHODS

**as.matrix** converts output to a matrix.

**plot** plots the output. Optional arguments are:

**ylab** Character string denoting the y-axis label for the plot. Default: `upperCase(attr(x, "type"))`.

**type** Line type (same as the `type` argument of the `par` function). Default: "l".

... Additional plot arguments (set internally by the `par` function).

**print** prints a summary of the output object.

### References

A. T. Walden, "Accurate Approximation of a 0th Order Discrete Prolate Spheroidal Sequence for Filtering and Data Tapering", *Signal Processing*, **18**, 341–8 (1989).

Percival, Donald B. and Constantine, William L. B. (2005) "Exact Simulation of Gaussian Time Series from Nonparametric Spectral Estimates with Application to Bootstrapping", *Journal of Computational and Graphical Statistics*, accepted for publication.

D.B. Percival and A. Walden (1993), *Spectral Analysis for Physical Applications: Multitaper and Conventional Univariate Techniques*, Cambridge University Press, Cambridge, UK.

S.K.Mitra, J. Kaiser (1993), *Handbook for Digital Signal Processing*, John Wiley and Sons, Inc.

### See Also

[taper](#).

### Examples

```
## change plot layout
gap <- 0.11
old.plt <- splitplot(4,4,1,gap=gap)

## create a plot of all supported tapers and
## windows
nms <- c("rectangle", "triangle", "raised cosine",
        "hanning", "hamming", "blackman",
        "nuttall", "gaussian", "kaiser",
        "chebyshev", "born jordan", "sine",
        "parzen", "papoulis", "daniell", "dpss")

for (i in seq(along=nms)){
  if (i > 1) splitplot(4,4,i,gap=gap)
  plot(taper(type=nms[i]))
}

## restore plot layout to initial state
par(old.plt)
```

# Index

\*Topic **ts**

taper, [7](#)

\*Topic **univar**

ACVS, [2](#)

SDF, [3](#)

ACVS, [2](#), [6](#)

as.matrix.SDF (SDF), [3](#)

as.matrix.taper (taper), [7](#)

plot.SDF (SDF), [3](#)

plot.taper (taper), [7](#)

print.SDF (SDF), [3](#)

print.taper (taper), [7](#)

SDF, [2](#), [3](#)

taper, [6](#), [7](#), [10](#)