

# Package ‘Rwave’

April 17, 2009

**Version** 1.24-2

**Date** 2008-05-20

**Title** Time-Frequency analysis of 1-D signals

**Author** S original by Rene Carmona <rcarmona@princeton.edu> and Bruno Torresani <bruno.torresani@cmi.univ-mrs.fr>; R port by Brandon Whitcher <brandon.j.whitcher@gsk.com>

**Maintainer** Brandon Whitcher <brandon.j.whitcher@gsk.com>

**Depends** R (>= 2.6.0)

**Description** Rwave is a library of R functions which provide an environment for the Time-Frequency analysis of 1-D signals (and especially for the wavelet and Gabor transforms of noisy signals). It was originally written for Splus by Rene Carmona, Bruno Torresani, and Wen L. Hwang, first at the University of California at Irvine and then at Princeton University. Credit should also be given to Andrea Wang whose functions on the dyadic wavelet transform are included. Rwave is based on the book: “Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S”, by Rene Carmona, Wen L. Hwang and Bruno Torresani, Academic Press, 1998.

**License** GPL (>= 2)

**URL** <http://www.orfe.princeton.edu/~rcarmona/>

**Repository** CRAN

**Date/Publication** 2008-05-21 07:21:12

## R topics documented:

A0 . . . . .	4
A4 . . . . .	4
adjust.length . . . . .	5
amber7 . . . . .	5
amber8 . . . . .	6
amber9 . . . . .	6

B0	7
B4	7
back1.000	8
back1.180	8
back1.220	9
C0	9
C4	10
cfamily	10
cgt	11
ch	12
check.maxresoln	13
cleanph	13
click	14
corona	14
coronoid	15
crc	17
crcrec	18
crfview	19
cwt	19
cwtimage	20
cwtp	21
cwtpolar	22
cwtsquiz	23
cwtTh	24
D0	25
D4	25
DOG	26
dwinverse	27
Ekg	27
epl	28
ext	28
fastgkernel	29
fastkernel	30
gabor	32
gcrec	33
gkernel	34
gregrec	35
gridrec	36
gsampleOne	37
gwave	38
gwave2	38
HOWAREYOU	39
hurst.est	40
icm	41
kernel	42
mbtrim	43
mntrim	44
morlet	45

morwave . . . . .	46
morwave2 . . . . .	46
mrecons . . . . .	47
mw . . . . .	48
noisywave . . . . .	49
npl . . . . .	49
plotResult . . . . .	50
plotwt . . . . .	50
purwave . . . . .	51
regrec . . . . .	51
regrec2 . . . . .	53
RidgeSampling . . . . .	54
ridrec . . . . .	55
rkernel . . . . .	56
srcrec . . . . .	57
skeleton . . . . .	58
skeleton2 . . . . .	59
smoothts . . . . .	60
smoothwt . . . . .	60
snake . . . . .	61
snakeview . . . . .	62
snakoid . . . . .	63
sridrec . . . . .	64
SVD . . . . .	65
tfgmax . . . . .	65
tflmax . . . . .	66
tfmean . . . . .	67
tfpct . . . . .	67
tfvar . . . . .	68
Undocumented . . . . .	69
vDOG . . . . .	69
vecgabor . . . . .	70
vecmorlet . . . . .	70
vgt . . . . .	71
vwt . . . . .	72
wpl . . . . .	72
wRidgeSampling . . . . .	73
wspec.pl . . . . .	74
WV . . . . .	74
YN . . . . .	75
YNdiff . . . . .	75
zerokernel . . . . .	76
zeroskeleton . . . . .	76
zeroskeleton2 . . . . .	77

---

A0

*Transient Signal*

---

**Description**

Transient signal.

**Usage**

data(A0)

**Format**

A vector containing 1024 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**

Carmona, R. A., W. L. Hwang and B Torresani (1998) *Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S*, Academic Press, San Diego.

---

A4

*Transient Signal*

---

**Description**

Transient signal.

**Usage**

data(A4)

**Format**

A vector containing 1024 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**

Carmona, R. A., W. L. Hwang and B Torresani (1998) *Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S*, Academic Press, San Diego.

---

adjust.length	<i>Zero Padding</i>
---------------	---------------------

---

**Description**

Add zeros to the end of the data if necessary so that its length is a power of 2. It returns the data with zeros added if necessary and the length of the adjusted data.

**Usage**

```
adjust.length(inputdata)
```

**Arguments**

inputdata     either a text file or an S object containing data.

**Value**

Zero-padded 1D array.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

---

amber7	<i>Pixel from Amber Camara</i>
--------	--------------------------------

---

**Description**

Pixel from amber camara.

**Usage**

```
data(amber7)
```

**Format**

A vector containing 7000 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**

Carmona, R. A., W. L. Hwang and B Torresani (1998) *Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S*, Academic Press, San Diego.

---

amber8

*Pixel from Amber Camara*

---

**Description**

Pixel from amber camara.

**Usage**

`data(amber8)`

**Format**

A vector containing 7000 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**

Carmona, R. A., W. L. Hwang and B Torresani (1998) *Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S*, Academic Press, San Diego.

---

amber9

*Pixel from Amber Camara*

---

**Description**

Pixel from amber camara.

**Usage**

`data(amber9)`

**Format**

A vector containing 7000 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**

Carmona, R. A., W. L. Hwang and B Torresani (1998) *Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S*, Academic Press, San Diego.

---

B0

*Transient Signal*

---

**Description**

Transient signal.

**Usage**

data(B0)

**Format**

A vector containing 1024 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**

Carmona, R. A., W. L. Hwang and B Torresani (1998) *Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S*, Academic Press, San Diego.

---

B4

*Transient Signal*

---

**Description**

Transient signal.

**Usage**

data(B4)

**Format**

A vector containing 1024 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**

Carmona, R. A., W. L. Hwang and B Torresani (1998) *Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S*, Academic Press, San Diego.

---

`back1.000`*Acoustic Returns*

---

**Description**

Acoustic returns from natural underwater clutter.

**Usage**

```
data(back1.000)
```

**Format**

A vector containing 7936 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**

Carmona, R. A., W. L. Hwang and B Torresani (1998) *Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S*, Academic Press, San Diego.

---

`back1.180`*Acoustic Returns*

---

**Description**

Acoustic returns from ...

**Usage**

```
data(back1.180)
```

**Format**

A vector containing 7936 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**

Carmona, R. A., W. L. Hwang and B Torresani (1998) *Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S*, Academic Press, San Diego.

---

`back1.220`*Acoustic Returns*

---

**Description**

Acoustic returns from an underwater metallic object.

**Usage**

```
data(back1.220)
```

**Format**

A vector containing 7936 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**

Carmona, R. A., W. L. Hwang and B Torresani (1998) *Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S*, Academic Press, San Diego.

---

`C0`*Transient Signal*

---

**Description**

Transient signal.

**Usage**

```
data(C0)
```

**Format**

A vector containing 1024 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**

Carmona, R. A., W. L. Hwang and B Torresani (1998) *Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S*, Academic Press, San Diego.

---

C4

*Transient Signal*

---

**Description**

Transient signal.

**Usage**

```
data(C4)
```

**Format**

A vector containing 1024 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**

Carmona, R. A., W. L. Hwang and B Torresani (1998) *Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S*, Academic Press, San Diego.

---

cfamily

*Ridge Chaining Procedure*

---

**Description**

Chains the ridge estimates produced by the function `crc`.

**Usage**

```
cfamily(ccridge, bstep=1, nbchain=100, ptile=0.05)
```

**Arguments**

<code>ccridge</code>	unchained ridge set as the output of the function <code>crc</code>
<code>bstep</code>	maximal length for a gap in a ridge.
<code>nbchain</code>	maximal number of chains produced by the function.
<code>ptile</code>	relative threshold for the ridges.

**Details**

`crc` returns a measure in time-frequency (or time-scale) space. `cfamily` turns it into a series of one-dimensional objects (ridges). The measure is first thresholded, with a relative threshold value set to the input parameter `ptile`. During the chaining procedure, gaps within a given ridge are allowed and filled in. The maximal length of such gaps is the input parameter `bstep`.

**Value**

Returns the results of the chaining algorithm

<code>ordered map</code>	image containing the ridges (displayed with different colors)
<code>chain</code>	2D array containing the chained ridges, according to the chain data structure <code>chain[,1]</code> : first point of the ridge <code>chain[,2]</code> : length of the chain <code>chain[,3:(chain[,2]+2)]</code> : values of the ridge
<code>nbchain</code>	number of chains produced by the algorithm

**References**

See discussion in text of “Practical Time-Frequency Analysis”.

**See Also**

`crc` for the ridge estimation, and `crcrec`, `gcrcrec` and `scrcrec` for corresponding reconstruction functions.

---

 cgt

*Continuous Gabor Transform*


---

**Description**

Computes the continuous Gabor transform with Gaussian window.

**Usage**

```
cgt(input, nvoice, freqstep=(1/nvoice), scale=1, plot=TRUE)
```

**Arguments**

<code>input</code>	input signal (possibly complex-valued).
<code>nvoice</code>	number of frequencies for which gabor transform is to be computed.
<code>freqstep</code>	Sampling rate for the frequency axis.
<code>scale</code>	Size parameter for the window.
<code>plot</code>	logical variable set to TRUE to display the modulus of the continuous gabor transform on the graphic device.

**Details**

The output contains the (complex) values of the gabor transform of the input signal. The format of the output is a 2D array (signal\_size x nb\_scales).

**Value**

continuous (complex) gabor transform (2D array).

**Warning**

freqstep must be less than 1/nvoice to avoid aliasing. freqstep=1/nvoice corresponds to the Nyquist limit.

**References**

See discussion in text of “Practical Time-Frequency Analysis”.

**See Also**

[cwt](#), [cwtp](#), [DOG](#) for continuous wavelet transforms. [cwtsqueeze](#) for synchrosqueezed wavelet transform.

---

ch

*Chen's Chirp*

---

**Description**

Chen's chirp.

**Usage**

```
data(ch)
```

**Format**

A vector containing 15,000 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**

Carmona, R. A., W. L. Hwang and B Torresani (1998) *Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S*, Academic Press, San Diego.

---

check.maxresoln      *Verify Maximum Resolution*

---

**Description**

Stop when  $2^{\text{maxresoln}}$  is larger than the signal size.

**Usage**

```
check.maxresoln(maxresoln, np)
```

**Arguments**

maxresoln	number of decomposition scales.
np	signal size.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[mw, mrecons.](#)

---

cleanph      *Threshold Phase based on Modulus*

---

**Description**

Sets to zero the phase of time-frequency transform when modulus is below a certain value.

**Usage**

```
cleanph(tfrep, thresh=0.01, plot=TRUE)
```

**Arguments**

tfrep	continuous time-frequency transform (2D array)
thresh	(relative) threshold.
plot	if set to TRUE, displays the maxima of cwt on the graphic device.

**Value**

thresholded phase (2D array)

**References**

See discussion in text of “Practical Time-Frequency Analysis”.

---

`click`*Dolphin Click Data*

---

**Description**

Dolphin click data.

**Usage**

```
data(click)
```

**Format**

A vector containing 2499 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**

Carmona, R. A., W. L. Hwang and B Torresani (1998) *Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S*, Academic Press, San Diego.

---

`corona`*Ridge Estimation by Corona Method*

---

**Description**

Estimate a (single) ridge from a time-frequency representation, using the corona method.

**Usage**

```
corona(tfrep, guess, tfspec=numeric(dim(tfrep)[2]), subrate=1,  
temprate=3, mu=1, lambda=2 * mu, iteration=1000000, seed=-7,  
stagnant=20000, costsub=1, plot=TRUE)
```

**Arguments**

<code>tfrep</code>	Time-Frequency representation (real valued).
<code>guess</code>	Initial guess for the algorithm.
<code>tfspec</code>	Estimate for the contribution of the noise to modulus.
<code>subrate</code>	Subsampling rate for ridge estimation.
<code>temprate</code>	Initial value of temperature parameter.

<code>mu</code>	Coefficient of the ridge's second derivative in cost function.
<code>lambda</code>	Coefficient of the ridge's derivative in cost function.
<code>iteration</code>	Maximal number of moves.
<code>seed</code>	Initialization of random number generator.
<code>stagnant</code>	Maximum number of stationary iterations before stopping.
<code>costsub</code>	Subsampling of cost function in output.
<code>plot</code>	When set(default), some results will be shown on the display.

### Details

To accelerate convergence, it is useful to preprocess modulus before running annealing method. Such a preprocessing (smoothing and subsampling of modulus) is implemented in `corona`. The parameter `subrate` specifies the subsampling rate.

### Value

Returns the estimated ridge and the cost function.

<code>ridge</code>	1D array (of same length as the signal) containing the ridge.
<code>cost</code>	1D array containing the cost function.

### Warning

The returned cost may be a large array, which is time consuming. The argument `costsub` allows subsampling the cost function.

### References

See discussion in text of "Practical Time-Frequency Analysis".

### See Also

`icm`, `coronoid`, `snake`, `snakoid`.

---

`coronoid`

*Ridge Estimation by Modified Corona Method*

---

### Description

Estimate a ridge using the modified corona method (modified cost function).

### Usage

```
coronoid(tfrep, guess, tfspec=numeric(dim(tfrep)[2]), subrate=1,
temprate=3, mu=1, lambda=2 * mu, iteration=1000000, seed=-7,
stagnant=20000, costsub=1, plot=TRUE)
```

**Arguments**

<code>tfrep</code>	Estimate for the contribution of the noise to modulus.
<code>guess</code>	Initial guess for the algorithm.
<code>tfspec</code>	Estimate for the contribution of the noise to modulus.
<code>subrate</code>	Subsampling rate for ridge estimation.
<code>temprate</code>	Initial value of temperature parameter.
<code>mu</code>	Coefficient of the ridge's derivative in cost function.
<code>lambda</code>	Coefficient of the ridge's second derivative in cost function.
<code>iteration</code>	Maximal number of moves.
<code>seed</code>	Initialization of random number generator.
<code>stagnant</code>	Maximum number of stationary iterations before stopping.
<code>costsub</code>	Subsampling of cost function in output.
<code>plot</code>	When set(default), some results will be shown on the display.

**Details**

To accelerate convergence, it is useful to preprocess modulus before running annealing method. Such a preprocessing (smoothing and subsampling of modulus) is implemented in `coronoid`. The parameter `subrate` specifies the subsampling rate.

**Value**

Returns the estimated ridge and the cost function.

<code>ridge</code>	1D array (of same length as the signal) containing the ridge.
<code>cost</code>	1D array containing the cost function.

**Warning**

The returned cost may be a large array. The argument `costsub` allows subsampling the cost function.

**References**

See discussion in text of “Practical Time-Frequency Analysis”.

**See Also**

`corona`, `icm`, `snake`, `snakoid`.

---

 crc

*Ridge Extraction by Crazy Climbers*


---

**Description**

Uses the "crazy climber algorithm" to detect ridges in the modulus of a continuous wavelet or a Gabor transform.

**Usage**

```
crc(tfrep, tfspec=numeric(dim(tfrep)[2]), bstep=3, iteration=10000,
rate=0.001, seed=-7, nbclimb=10, flag.int=TRUE, chain=TRUE,
flag.temp=FALSE)
```

**Arguments**

tfrep	modulus of the (wavelet or Gabor) transform.
tfspec	numeric vector which gives, for each value of the scale or frequency the expected size of the noise contribution.
bstep	stepsize for random walk of the climbers.
iteration	number of iterations.
rate	initial value of the temperature.
seed	initial value of the random number generator.
nbclimb	number of crazy climbers.
flag.int	if set to TRUE, the weighted occupation measure is computed.
chain	if set to TRUE, chaining of the ridges is done.
flag.temp	if set to TRUE: constant temperature.

**Value**

Returns a 2D array called beemap containing the (weighted or unweighted) occupation measure (integrated with respect to time)

**References**

See discussion in text of "Practical Time-Frequency Analysis".

**See Also**

[corona](#), [icm](#), [coronoid](#), [snake](#), [snakoid](#) for ridge estimation, [cfamily](#) for chaining and [crrc](#), [grrc](#), [srrc](#) for reconstruction.

crcrec

*Crazy Climbers Reconstruction by Penalization***Description**

Reconstructs a real valued signal from the output of `crc` (wavelet case) by minimizing an appropriate quadratic form.

**Usage**

```
crcrec(sinput, inputwt, beemap, noct, nvoice, compr, minnbnodes=2,
w0=2 * pi, bstep=5, ptile=0.01, epsilon=0, fast=FALSE, para=5, real=FALSE,
plot=2)
```

**Arguments**

<code>sinput</code>	original signal.
<code>inputwt</code>	wavelet transform.
<code>beemap</code>	occupation measure, output of <code>crc</code> .
<code>noct</code>	number of octaves.
<code>nvoice</code>	number of voices per octave.
<code>compr</code>	compression rate for sampling the ridges.
<code>minnbnodes</code>	minimal number of points per ridge.
<code>w0</code>	center frequency of the wavelet.
<code>bstep</code>	size (in the time direction) of the steps for chaining.
<code>ptile</code>	relative threshold of occupation measure.
<code>epsilon</code>	constant in front of the smoothness term in penalty function.
<code>fast</code>	if set to TRUE, uses trapezoidal rule to evaluate $Q_2$ .
<code>para</code>	scale parameter for extrapolating the ridges.
<code>real</code>	if set to TRUE, uses only real constraints.
<code>plot</code>	1: displays signal, components, and reconstruction one after another. 2: displays signal, components and reconstruction.

**Details**

When `ptile` is high, boundary effects may appear. `para` controls extrapolation of the ridge.

**Value**

Returns a structure containing the following elements:

<code>rec</code>	reconstructed signal.
<code>ordered</code>	image of the ridges (with different colors).
<code>comp</code>	2D array containing the signals reconstructed from ridges.

**See Also**

[crc](#), [cfamily](#), [srcrec](#).

---

crfview

*Display chained ridges*

---

**Description**

displays a family of chained ridges, output of [cfamily](#).

**Usage**

```
crfview(beemap, twod=TRUE)
```

**Arguments**

beemap	Family of chained ridges, output of <a href="#">cfamily</a> .
twod	If set to T, displays the ridges as an image. If set to F, displays as a series of curves.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[crc](#), [cfamily](#) for crazy climbers and corresponding chaining algorithms.

---

cwt

*Continuous Wavelet Transform*

---

**Description**

Computes the continuous wavelet transform with for the (complex-valued) Morlet wavelet.

**Usage**

```
cwt(input, noctave, nvoice=1, w0=2 * pi, twod=TRUE, plot=TRUE)
```

**Arguments**

<code>input</code>	input signal (possibly complex-valued)
<code>noctave</code>	number of powers of 2 for the scale variable
<code>nvoice</code>	number of scales in each octave (i.e. between two consecutive powers of 2).
<code>w0</code>	central frequency of the wavelet.
<code>twoD</code>	logical variable set to T to organize the output as a 2D array (signal_size x nb_scales), otherwise, the output is a 3D array (signal_size x octave x nvoice).
<code>plot</code>	if set to T, display the modulus of the continuous wavelet transform on the graphic device.

**Details**

The output contains the (complex) values of the wavelet transform of the input signal. The format of the output can be

2D array (signal\_size x nb\_scales)

3D array (signal\_size x octave x nvoice)

Since Morlet's wavelet is not strictly speaking a wavelet (it is not of vanishing integral), artifacts may occur for certain signals.

**Value**

continuous (complex) wavelet transform

**References**

See discussions in the text of "Practical Time-Frequency Analysis".

**See Also**

[cwtP](#), [cwtTh](#), [DOG](#), [gabor](#).

---

`cwtimage`*Continuous Wavelet Transform Display*

---

**Description**

Converts the output (modulus or argument) of `cwtPolar` to a 2D array and displays on the graphic device.

**Usage**

```
cwtimage(input)
```

**Arguments**

`input`                    3D array containing a continuous wavelet transform

**Details**

The output contains the (complex) values of the wavelet transform of the input signal. The format of the output can be

2D array (signal\_size x nb\_scales)

3D array (signal\_size x noctave x nvoice)

**Value**

2D array continuous (complex) wavelet transform

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[cwtPolar](#), [cwt](#), [DOG](#).

---

cwtP

*Continuous Wavelet Transform with Phase Derivative*

---

**Description**

Computes the continuous wavelet transform with (complex-valued) Morlet wavelet and its phase derivative.

**Usage**

```
cwtP(input, noctave, nvoice=1, w0=2 * pi, twoD=TRUE, plot=TRUE)
```

**Arguments**

`input`                    input signal (possibly complex-valued)

`noctave`                number of powers of 2 for the scale variable

`nvoice`                 number of scales in each octave (i.e., between two consecutive powers of 2).

`w0`                     central frequency of the wavelet.

`twoD`                  logical variable set to `T` to organize the output as a 2D array (signal size × nb scales), otherwise, the output is a 3D array (signal size × noctave × nvoice).

`plot`                  if set to `T`, display the modulus of the continuous wavelet transform on the graphic device.

**Value**

list containing the continuous (complex) wavelet transform and the phase derivative

`wt`                    array of complex numbers for the values of the continuous wavelet transform.  
`f`                        array of the same dimensions containing the values of the derivative of the phase of the continuous wavelet transform.

**References**

See discussions in the text of "Practical Time-Frequency Analysis".

**See Also**

[cgt](#), [cwt](#), [cwtTh](#), [DOG](#) for wavelet transform, and [gabor](#) for continuous Gabor transform.

---

`cwtpolar`

*Conversion to Polar Coordinates*

---

**Description**

Converts one of the possible outputs of the function [cwt](#) to modulus and phase.

**Usage**

```
cwtpolar(cwt, threshold=0)
```

**Arguments**

`cwt`                    3D array containing the values of a continuous wavelet transform in the format (signal size  $\times$  octave  $\times$  nvoice) as in the output of the function [cwt](#) with the logical flag `twodimension` set to `FALSE`.  
`threshold`            value of a level for the absolute value of the modulus below which the value of the argument of the output is set to  $-\pi$ .

**Details**

The output contains the (complex) values of the wavelet transform of the input signal. The format of the output can be

2D array (signal size  $\times$  nb\_scales)

3D array (signal size  $\times$  octave  $\times$  nvoice)

**Value**

Modulus and Argument of the values of the continuous wavelet transform

`output1`            3D array giving the values (in the same format as the input) of the modulus of the input.  
`output2`            3D array giving the values of the argument of the input.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[cwt](#), [DOG](#), [cwtimage](#).

---

cwtssquiz

*Squeezed Continuous Wavelet Transform*

---

**Description**

Computes the synchrosqueezed continuous wavelet transform with the (complex-valued) Morlet wavelet.

**Usage**

```
cwtssquiz(input, noctave, nvoice=1, w0=2 * pi, twoD=TRUE, plot=TRUE)
```

**Arguments**

<code>input</code>	input signal (possibly complex-valued)
<code>noctave</code>	number of powers of 2 for the scale variable
<code>nvoice</code>	number of scales in each octave (i.e. between two consecutive powers of 2).
<code>w0</code>	central frequency of the wavelet.
<code>twoD</code>	logical variable set to T to organize the output as a 2D array (signal size $\times$ nb scales), otherwise, the output is a 3D array (signal size $\times$ noctave $\times$ nvoice).
<code>plot</code>	logical variable set to T to T to display the modulus of the squeezed wavelet transform on the graphic device.

**Details**

The output contains the (complex) values of the squeezed wavelet transform of the input signal. The format of the output can be  
 2D array (signal size  $\times$  nb scales),  
 3D array (signal size  $\times$  noctave  $\times$  nvoice).

**Value**

synchrosqueezed continuous (complex) wavelet transform

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[cwt](#), [cwtp](#), [DOG](#), [cgt](#).

---

`cwtTh`*Cauchy's wavelet transform*

---

**Description**

Compute the continuous wavelet transform with (complex-valued) Cauchy's wavelet.

**Usage**

```
cwtTh(input, noctave, nvoice=1, moments, twoD=TRUE, plot=TRUE)
```

**Arguments**

<code>input</code>	input signal (possibly complex-valued).
<code>noctave</code>	number of powers of 2 for the scale variable.
<code>nvoice</code>	number of scales in each octave (i.e. between two consecutive powers of 2).
<code>moments</code>	number of vanishing moments.
<code>twoD</code>	logical variable set to T to organize the output as a 2D array (signal size x nb scales), otherwise, the output is a 3D array (signal size x noctave x nvoice).
<code>plot</code>	if set to T, display the modulus of the continuous wavelet transform on the graphic device.

**Details**

The output contains the (complex) values of the wavelet transform of the input signal. The format of the output can be

2D array (signal size  $\times$  nb scales)

3D array (signal size  $\times$  noctave  $\times$  nvoice)

**Value**

`tmp` continuous (complex) wavelet transform.

**References**

See discussions in the text of "Practical Time-Frequency Analysis".

**See Also**

[cwt](#), [cwtp](#), [DOG](#), [gabor](#).

---

D0

*Transient Signal*

---

**Description**

Transient signal.

**Usage**

data(D0)

**Format**

A vector containing 1024 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**

Carmona, R. A., W. L. Hwang and B Torresani (1998) *Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S*, Academic Press, San Diego.

---

D4

*Transient Signal*

---

**Description**

Transient signal.

**Usage**

data(D4)

**Format**

A vector containing 1024 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**

Carmona, R. A., W. L. Hwang and B Torresani (1998) *Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S*, Academic Press, San Diego.

**Description**

Computes the continuous wavelet transform with for (complex-valued) derivative of Gaussian wavelets.

**Usage**

```
DOG(input, noctave, nvoice=1, moments, twoD=TRUE, plot=TRUE)
```

**Arguments**

<code>input</code>	input signal (possibly complex-valued).
<code>noctave</code>	number of powers of 2 for the scale variable.
<code>moments</code>	number of vanishing moments of the wavelet (order of the derivative).
<code>nvoice</code>	number of scales in each octave (i.e. between two consecutive powers of 2)
<code>twoD</code>	logical variable set to T to organize the output as a 2D array ( <code>signal_size</code> x <code>nb_scales</code> ), otherwise, the output is a 3D array ( <code>signal_size</code> x <code>noctave</code> x <code>nvoice</code> )
<code>plot</code>	if set to T, display the modulus of the continuous wavelet transform on the graphic device

**Details**

The output contains the (complex) values of the wavelet transform of the input signal. The format of the output can be

2D array (`signal_size` x `nb_scales`)

3D array (`signal_size` x `noctave` x `nvoice`)

**Value**

continuous (complex) wavelet transform

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[cwt](#), [cwtp](#), [cwtsquiz](#), [cgt](#).

---

dwinverse                      *Inverse Dyadic Wavelet Transform*

---

**Description**

Invert the dyadic wavelet transform.

**Usage**

```
dwinverse(wt, filtername="Gaussian1")
```

**Arguments**

wt	dyadic wavelet transform
filtername	filters used. ("Gaussian1" stands for the filters corresponds to those of Mallat and Zhong's wavlet. And "Haar" stands for the filters of Haar basis.

**Value**

Reconstructed signal

**References**

See discussions in the text of "Practical Time-Frequency Analysis".

**See Also**

[mw](#), [ext](#), [mrecons](#).

---

Ekg                                      *Heart Rate Data*

---

**Description**

Successive beat-to-beat intervals for a normal patient.

**Usage**

```
data(Ekg)
```

**Format**

A vector containing 16,042 observations.

**Source**

See discussions in the text of "Practical Time-Frequency Analysis".

**References**

Carmona, R. A., W. L. Hwang and B Torresani (1998) *Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S*, Academic Press, San Diego.

---

epl *Plot Dyadic Wavelet Transform Extrema*

---

**Description**

Plot dyadic wavelet transform extrema (output of [ext](#)).

**Usage**

```
epl(dwext)
```

**Arguments**

dwext            dyadic wavelet transform (output of [ext](#)).

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[mw](#), [ext](#), [wpl](#).

---

ext *Extrema of Dyadic Wavelet Transform*

---

**Description**

Compute the local extrema of the dyadic wavelet transform modulus.

**Usage**

```
ext(wt, scale=FALSE, plot=TRUE)
```

**Arguments**

wt                dyadic wavelet transform.  
scale             flag indicating if the extrema at each resolution will be plotted at the same scale.  
plot              if set to TRUE, displays the transform on the graphics device.

**Value**

Structure containing:

original	original signal.
extrema	extrema representation.
Sf	coarse resolution of signal.
maxresoln	number of decomposition scales.
np	size of signal.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[mw](#), [mrecons](#).

---

fastgkernel

*Kernel for Reconstruction from Gabor Ridges*

---

**Description**

Computes the cost from the sample of points on the estimated ridge and the matrix used in the reconstruction of the original signal, using simple trapezoidal rule for integrals.

**Usage**

```
fastgkernel(node, phinode, freqstep, scale, x.inc=1, x.min=node[1],
x.max=node[length(node)], plot=FALSE)
```

**Arguments**

node	values of the variable $b$ for the nodes of the ridge
phinode	values of the frequency variable $\omega$ for the nodes of the ridge
freqstep	sampling rate for the frequency axis
scale	size of the window
x.inc	step unit for the computation of the kernel.
x.min	minimal value of $x$ for the computation of $G_2$ .
x.max	maximal value of $x$ for the computation of $G_2$ .
plot	if set to TRUE, displays the modulus of the matrix of $G_2$ .

**Details**

Uses trapezoidal rule (instead of Romberg’s method) to evaluate the kernel.

**Value**

matrix of the  $G_2$  kernel.

**References**

See discussions in the text of “Time-Frequency Analysis”.

**See Also**

[gkernel](#), [fastkernel](#), [rkernel](#), [zerokernel](#).

---

fastkernel

*Kernel for Reconstruction from Wavelet Ridges*

---

**Description**

Computes the cost from the sample of points on the estimated ridge and the matrix used in the reconstruction of the original signal, using simple trapezoidal rule for integrals.

**Usage**

```
fastkernel(node, phinode, nvoice, x.inc=1, x.min=node[1],
           x.max=node[length(node)], w0=2 * pi, plot=FALSE)
```

**Arguments**

node	values of the variable b for the nodes of the ridge.
phinode	values of the scale variable a for the nodes of the ridge.
nvoice	number of scales within 1 octave.
x.inc	step unit for the computation of the kernel
x.min	minimal value of x for the computation of $Q_2$ .
x.max	maximal value of x for the computation of $Q_2$ .
w0	central frequency of the wavelet
plot	if set to TRUE, displays the modulus of the matrix of $Q_2$ .

**Details**

Uses trapezoidal rule (instead of Romberg’s method) to evaluate the kernel.

**Value**

matrix of the  $Q_2$  kernel.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[kernel](#), [rkernel](#), [gkernel](#), [zerokernel](#).

**Examples**

```
# The function is currently defined as
function(node, phinode, nvoice, x.inc = 1, x.min = node[1], x.max = node[length(node)], w0 =
{
#####
#   fastkernel:
#   -----
#       Same as kernel, except that the kernel is computed
#       using Riemann sums instead of Romberg integration.
#
#   Input:
#   -----
#       node: values of the variable b for the nodes of the ridge
#       phinode: values of the scale variable a for the nodes of the ridge
#       nvoice: number of scales within 1 octave
#       x.inc: step unit for the computation of the kernel
#       x.min: minimal value of x for the computation of Q2
#       x.max: maximal value of x for the computation of Q2
#       w0: central frequency of the wavelet
#       plot: if set to TRUE, displays the modulus of the matrix of Q2
#
#   Output:
#   -----
#       ker: matrix of the Q2 kernel
#
#####
  lng <- as.integer((x.max - x.min)/x.inc) + 1
  nbnode <- length(node)
  b.start <- x.min - 50
  b.end <- x.max + 50
  ker.r <- matrix(0, lng, lng)
  ker.i <- matrix(0, lng, lng)
  dim(ker.r) <- c(lng * lng, 1)
  dim(ker.i) <- c(lng * lng, 1)
  phinode <- 2 * 2^(phinode/nvoice)
  z <- .C(fastkernel,
         ker.r = as.double(ker.r),
         ker.i = as.double(ker.i),
         as.integer(x.min),
         as.integer(x.max),
         as.integer(x.inc),
         as.integer(lng),
         as.double(node),
         as.double(phinode),
         as.integer(nbnode),
         as.double(w0),
         as.double(b.start),
         as.double(b.end))
}
```

```
ker.r <- z$ker.r
ker.i <- z$ker.i
dim(ker.r) <- c(lng, lng)
dim(ker.i) <- c(lng, lng)
ker <- matrix(0, lng, lng)
i <- sqrt(as.complex(-1))
ker <- ker.r + i * ker.i
if(plot == T) {
  par(mfrow = c(1, 1))
  image(Mod(ker))
  title("Matrix of the reconstructing kernel (modulus)")
}
ker
}
```

---

gabor

*Generate Gabor function*

---

### Description

Generates a Gabor for given location and frequency.

### Usage

```
gabor(sigsize, location, frequency, scale)
```

### Arguments

sigsize	length of the Gabor function.
location	position of the Gabor function.
frequency	frequency of the Gabor function.
scale	size parameter for the Gabor function.

### Value

complex 1D array of size sigsize.

### References

See discussions in the text of “Practical Time-Frequency Analysis”.

### See Also

[morlet](#).

**Description**

Reconstructs a real valued signal from ridges found by crazy climbers on a gabor transform.

**Usage**

```
gcrcrec(siginput, inputgt, beemap, nvoice, freqstep, scale, compr,
bstep=5, ptile=0.01, epsilon=0, fast=TRUE, para=5, minnbnodes=3,
hflag=FALSE, real=FALSE, plot=2)
```

**Arguments**

siginput	original signal.
inputgt	Gabor transform.
beemap	occupation measure, output of <code>crc</code> .
nvoice	number of frequencies.
freqstep	sampling step for frequency axis.
scale	size of windows.
compr	compression rate to be applied to the ridges.
bstep	size (in the time direction) of the steps for chaining.
ptile	threshold of ridge
epsilon	constant in front of the smoothness term in penalty function.
fast	if set to TRUE, uses trapezoidal rule to evaluate $Q_2$ .
para	scale parameter for extrapolating the ridges.
minnbnodes	minimal number of points per ridge.
hflag	if set to FALSE, uses the identity as first term in the kernel. If not, uses $Q_1$ instead.
real	if set to TRUE, uses only real constraints.
plot	
1	displays signal, components, and reconstruction one after another.
2	displays signal, components and reconstruction.

**Details**

When ptile is high, boundary effects may appear. para controls extrapolation of the ridge.

**Value**

Returns a structure containing the following elements:

rec	reconstructed signal.
ordered	image of the ridges (with different colors).
comp	2D array containing the signals reconstructed from ridges.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[crc](#), [cfamily](#), [csrcrec](#), [sccrec](#).

---

gkernel

*Kernel for Reconstruction from Gabor Ridges*


---

**Description**

Computes the cost from the sample of points on the estimated ridge and the matrix used in the reconstruction of the original signal.

**Usage**

```
gkernel(node, phinode, freqstep, scale, x.inc=1, x.min=node[1],
x.max=node[length(node)], plot=FALSE)
```

**Arguments**

node	values of the variable b for the nodes of the ridge.
phinode	values of the scale variable a for the nodes of the ridge.
freqstep	sampling rate for the frequency axis.
scale	size of the window.
x.inc	step unit for the computation of the kernel.
x.min	minimal value of x for the computation of $Q_2$ .
x.max	maximal value of x for the computation of $Q_2$ .
plot	if set to TRUE, displays the modulus of the matrix of $Q_2$ .

**Value**

matrix of the  $Q_2$  kernel

**References**

See discussions in the text of “Time-Frequency Analysis”.

**See Also**

[fastgkernel](#), [kernel](#), [rkernel](#), [fastkernel](#), [zerokernel](#).

---

gregrec

*Reconstruction from a Ridge*

---

**Description**

Reconstructs signal from a “regularly sampled” ridge, in the Gabor case.

**Usage**

```
gregrec(sinput, ginput, phi, nbnodes, nvoice, freqstep, scale,
epsilon=0, fast=FALSE, plot=FALSE, para=0, hflag=FALSE, real=FALSE,
check=FALSE)
```

**Arguments**

sinput	input signal.
ginput	Gabor transform, output of <a href="#">cgt</a> .
phi	unsampled ridge.
nbnodes	number of nodes used for the reconstruction.
nvoice	number of different scales per octave
freqstep	sampling rate for the frequency axis
scale	size parameter for the Gabor function.
epsilon	coefficient of the $Q_2$ term in reconstruction kernel
fast	if set to T, the kernel is computed using trapezoidal rule.
plot	if set to TRUE, displays original and reconstructed signals
para	scale parameter for extrapolating the ridges.
hflag	if set to TRUE, uses $Q_1$ as first term in the kernel.
real	if set to TRUE, uses only real constraints on the transform.
check	if set to TRUE, computes <a href="#">cwt</a> of reconstructed signal.

**Value**

Returns a list containing:

sol	reconstruction from a ridge.
A	<gaborlets,dualgaborlets> matrix.
lam	coefficients of dual wavelets in reconstructed signal.
dualwave	array containing the dual wavelets.
gaborets	array containing the wavelets on sampled ridge.
solskel	Gabor transform of sol, restricted to the ridge.
inputskel	Gabor transform of signal, restricted to the ridge.
Q2	second part of the reconstruction kernel.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[regrec](#).

---

gridrec

*Reconstruction from a Ridge*

---

**Description**

Reconstructs signal from sample of a ridge, in the Gabor case.

**Usage**

```
gridrec(gtinput, node, phinode, nvoice, freqstep, scale, Qin,
epsilon, np, real=FALSE, check=FALSE)
```

**Arguments**

gtinput	Gabor transform, output of <a href="#">cgt</a> .
node	time coordinates of the ridge samples.
phinode	frequency coordinates of the ridge samples.
nvoice	number of different frequencies.
freqstep	sampling rate for the frequency axis.
scale	scale of the window.
Qinv	inverse of the matrix $Q$ of the quadratic form.
epsilon	coefficient of the $Q_2$ term in reconstruction kernel
np	number of samples of the reconstructed signal.
real	if set to TRUE, uses only constraints on the real part of the transform.
check	if set to TRUE, computes <a href="#">cgt</a> of reconstructed signal.

**Value**

Returns a list containing the reconstructed signal and the chained ridges.

sol	reconstruction from a ridge.
A	<gaborlets,dualgaborlets> matrix.
lam	coefficients of dual gaborlets in reconstructed signal.
dualwave	array containing the dual gaborlets.
gaboret	array of gaborlets located on the ridge samples.
solskel	Gabor transform of sol, restricted to the ridge.
inputskel	Gabor transform of signal, restricted to the ridge.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[sridrec](#), [gregrec](#), [regrec](#), [regrec2](#).

---

gsampleOne

*Sampled Identity*

---

**Description**

Generate a sampled identity matrix.

**Usage**

```
gsampleOne(node, scale, np)
```

**Arguments**

node	location of the reconstruction gabor functions.
scale	scale of the gabor functions.
np	size of the reconstructed signal.

**Value**

diagonal of the “sampled”  $Q_1$  term (1D vector)

**References**

See discussions in the text of “Time-Frequency Analysis”.

**See Also**

[kernel](#), [gkernel](#).

---

 gwave

*Gabor Functions on a Ridge*


---

**Description**

Generation of Gabor functions located on the ridge.

**Usage**

```
gwave(bridge, omegaridge, nvoice, freqstep, scale, np, N)
```

**Arguments**

bridge	time coordinates of the ridge samples
omegaridge	frequency coordinates of the ridge samples
nvoice	number of different scales per octave
freqstep	sampling rate for the frequency axis
scale	scale of the window
np	size of the reconstruction kernel
N	number of complex constraints

**Value**

Array of Gabor functions located on the ridge samples

**References**

See discussions in the text of "Time-Frequency Analysis".

**See Also**

[gwave2](#), [morwave](#), [morwave2](#).

---

 gwave2

*Real Gabor Functions on a Ridge*


---

**Description**

Generation of the real parts of gabor functions located on a ridge. (modification of [gwave](#).)

**Usage**

```
gwave2(bridge, omegaridge, nvoice, freqstep, scale, np, N)
```

**Arguments**

bridge	time coordinates of the ridge samples
omegaridge	frequency coordinates of the ridge samples
nvoice	number of different scales per octave
freqstep	sampling rate for the frequency axis
scale	scale of the window
np	size of the reconstruction kernel
N	number of complex constraints

**Value**

Array of real Gabor functions located on the ridge samples

**References**

See discussions in the text of “Time-Frequency Analysis”.

**See Also**

[gwave](#), [morwave](#), [morwave2](#).

---

HOWAREYOU

*How Are You?*

---

**Description**

Example of speech signal.

**Usage**

```
data (HOWAREYOU)
```

**Format**

A vector containing 5151 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**

Carmona, R. A., W. L. Hwang and B Torresani (1998) *Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S*, Academic Press, San Diego.

hurst.est

*Estimate Hurst Exponent***Description**

Estimates Hurst exponent from a wavelet transform.

**Usage**

```
hurst.est(wspec, range, nvoice, plot=TRUE)
```

**Arguments**

wspec	wavelet spectrum (output of <code>tfmean</code> )
range	range of scales from which estimate the exponent.
nvoice	number of scales per octave of the wavelet transform.
plot	if set to TRUE, displays regression line on current plot.

**Value**

complex 1D array of size sigsize.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[tfmean](#), [wspec.pl](#).

**Examples**

```
# White Noise Hurst Exponent: The plots on the top row of Figure 6.8
# were produced by the following S-commands. These make use of the two
# functions Hurst.est (estimation of Hurst exponent from CWT) and
# wspec.pl (display wavelet spectrum).

# Compare the periodogram and the wavelet spectral estimate.
wnoise <- rnorm(8192)
plot.ts(wnoise)
spwnoise <- fft(wnoise)
spwnoise <- Mod(spwnoise)
spwnoise <- spwnoise*spwnoise
plot(spwnoise[1:4096], log="xy", type="l")
lswnoise <- lsfit(log10(1:4096), log10(spwnoise[1:4096]))
abline(lswnoise$coef)
cwtwnoise <- DOG(wnoise, 10, 5, 1, plot=FALSE)
mcwtwnoise <- Mod(cwtwnoise)
```

```
mcwtwnoise <- mcwtwnoise*mcwtwnoise
wspwnoise <- tfmean(mcwtwnoise, plot=FALSE)
wspec.pl(wspwnoise, 5)
hurst.est(wspwnoise, 1:50, 5)
```

icm

*Ridge Estimation by ICM Method***Description**

Estimate a (single) ridge from a time-frequency representation, using the ICM minimization method.

**Usage**

```
icm(modulus, guess, tfspec=numeric(dim(modulus)[2]), subrate=1,
mu=1, lambda=2 * mu, iteration=100)
```

**Arguments**

<code>modulus</code>	Time-Frequency representation (real valued).
<code>guess</code>	Initial guess for the algorithm.
<code>tfspec</code>	Estimate for the contribution of the noise to modulus.
<code>subrate</code>	Subsampling rate for ridge estimation.
<code>mu</code>	Coefficient of the ridge's second derivative in cost function.
<code>lambda</code>	Coefficient of the ridge's derivative in cost function.
<code>iteration</code>	Maximal number of moves.

**Details**

To accelerate convergence, it is useful to preprocess modulus before running annealing method. Such a preprocessing (smoothing and subsampling of modulus) is implemented in `icm`. The parameter `subrate` specifies the subsampling rate.

**Value**

Returns the estimated ridge and the cost function.

<code>ridge</code>	1D array (of same length as the signal) containing the ridge.
<code>cost</code>	1D array containing the cost function.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[corona](#), [coronoid](#), and [snake](#), [snakoid](#).

---

`kernel`*Kernel for Reconstruction from Wavelet Ridges*

---

**Description**

Computes the cost from the sample of points on the estimated ridge and the matrix used in the reconstruction of the original signal

**Usage**

```
kernel(node, phinode, nvoice, x.inc=1, x.min=node[1],  
x.max=node[length(node)], w0=2 * pi, plot=FALSE)
```

**Arguments**

<code>node</code>	values of the variable $b$ for the nodes of the ridge.
<code>phinode</code>	values of the scale variable $a$ for the nodes of the ridge.
<code>nvoice</code>	number of scales within 1 octave.
<code>x.inc</code>	step unit for the computation of the kernel.
<code>x.min</code>	minimal value of $x$ for the computation of $Q_2$ .
<code>x.max</code>	maximal value of $x$ for the computation of $Q_2$ .
<code>w0</code>	central frequency of the wavelet.
<code>plot</code>	if set to TRUE, displays the modulus of the matrix of $Q_2$ .

**Details**

The kernel is evaluated using Romberg's method.

**Value**

matrix of the  $Q_2$  kernel

**References**

See discussions in the text of "Time-Frequency Analysis".

**See Also**

[gkernel](#), [rkernel](#), [zerokernel](#).

---

`mbtrim`*Trim Dyadic Wavelet Transform Extrema*

---

**Description**

Trimming of dyadic wavelet transform local extrema, using bootstrapping.

**Usage**

```
mbtrim(extrema, scale=FALSE, prct=0.95)
```

**Arguments**

<code>extrema</code>	dyadic wavelet transform extrema (output of <a href="#">ext</a> ).
<code>scale</code>	when set, the wavelet transform at each scale will be plotted with the same scale.
<code>prct</code>	

**Details**

The distribution of extrema of dyadic wavelet transform at each scale is generated by bootstrap method, and the 95% critical value is used for thresholding the extrema of the signal.

**Value**

Structure containing

<code>original</code>	original signal.
<code>extrema</code>	trimmed extrema representation.
<code>Sf</code>	coarse resolution of signal.
<code>maxresoln</code>	number of decomposition scales.
<code>np</code>	size of signal.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[mnttrim](#), [mrecons](#), [ext](#).

---

`mntrim`*Trim Dyadic Wavelet Transform Extrema*

---

### Description

Trimming of dyadic wavelet transform local extrema, assuming normal distribution.

### Usage

```
mntrim(extrema, scale=FALSE, prct=0.95)
```

### Arguments

<code>extrema</code>	dyadic wavelet transform extrema (output of <a href="#">ext</a> ).
<code>scale</code>	when set, the wavelet transform at each scale will be plotted with the same scale.
<code>prct</code>	

### Details

The distribution of extrema of dyadic wavelet transform at each scale is generated by simulation, assuming a normal distribution, and the 95% critical value is used for thresholding the extrema of the signal.

### Value

Structure containing

<code>original</code>	original signal.
<code>extrema</code>	trimmed extrema representation.
<code>Sf</code>	coarse resolution of signal.
<code>maxresoln</code>	number of decomposition scales.
<code>np</code>	size of signal.

### References

See discussions in the text of “Practical Time-Frequency Analysis”.

### See Also

[mbtrim](#), [mrecons](#), [ext](#).

---

`morlet`*Morlet Wavelets*

---

**Description**

Computes a Morlet wavelet at the point of the time-scale plane given in the input

**Usage**

```
morlet(sigsize, location, scale, w0=2 * pi)
```

**Arguments**

<code>sigsize</code>	length of the output.
<code>location</code>	time location of the wavelet.
<code>scale</code>	scale of the wavelet.
<code>w0</code>	central frequency of the wavelet.

**Details**

The details of this construction (including the definition formulas) are given in the text.

**Value**

Returns the values of the complex Morlet wavelet at the point of the time-scale plane given in the input

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[gabor](#).

---

 morwave

*Ridge Morvelets*


---

**Description**

Generates the Morlet wavelets at the sample points of the ridge.

**Usage**

```
morwave(bridge, aridge, nvoice, np, N, w0=2 * pi)
```

**Arguments**

bridge	time coordinates of the ridge samples.
aridge	scale coordinates of the ridge samples.
nvoice	number of different scales per octave.
np	number of samples in the input signal.
N	size of reconstructed signal.
w0	central frequency of the wavelet.

**Value**

Returns the Morlet wavelets at the samples of the time-scale plane given in the input: complex array of Morlet wavelets located on the ridge samples

**References**

See discussions in the text of “Time-Frequency Analysis”.

**See Also**

[morwave2](#), [gwave](#), [gwave2](#).

---

 morwave2

*Real Ridge Morvelets*


---

**Description**

Generates the real parts of the Morlet wavelets at the sample points of a ridge

**Usage**

```
morwave2(bridge, aridge, nvoice, np, N, w0=2 * pi)
```

**Arguments**

bridge	time coordinates of the ridge samples.
aridge	scale coordinates of the ridge samples.
nvoice	number of different scales per octave.
np	number of samples in the input signal.
N	size of reconstructed signal.
w0	central frequency of the wavelet.

**Value**

Returns the real parts of the Morlet wavelets at the samples of the time-scale plane given in the input: array of Morlet wavelets located on the ridge samples

**References**

See discussions in the text of “Time-Frequency Analysis”.

**See Also**

[morwave](#), [gwave](#), [gwave2](#).

---

mrecons

---

*Reconstruct from Dyadic Wavelet Transform Extrema*


---

**Description**

Reconstruct from dyadic wavelet transform modulus extrema. The reconstructed signal preserves locations and values at extrema.

**Usage**

```
mrecons(extrema, filtername="Gaussian1", readflag=TRUE)
```

**Arguments**

extrema	the extrema representation.
filtername	filter used for dyadic wavelet transform.
readflag	if set to T, read reconstruction kernel from precomputed file.

**Details**

The reconstruction involves only the wavelet coefficients, without taking care of the coarse scale component. The latter may be added a posteriori.

**Value**

Structure containing

f	the reconstructed signal.
g	reconstructed signal plus mean of original signal.
h	reconstructed signal plus coarse scale component of original signal.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[mw](#), [ext](#).

---

 mw

*Dyadic Wavelet Transform*

---

**Description**

Dyadic wavelet transform, with Mallat’s wavelet. The reconstructed signal preserves locations and values at extrema.

**Usage**

```
mw(inputdata, maxresoln, filtername="Gaussian1", scale=FALSE, plot=TRUE)
```

**Arguments**

inputdata	either a text file or an R object containing data.
maxresoln	number of decomposition scales.
filtername	name of filter (either Gaussian1 for Mallat and Zhong’s wavelet or Haar wavelet).
scale	when set, the wavelet transform at each scale is plotted with the same scale.
plot	indicate if the wavelet transform at each scale will be plotted.

**Details**

The decomposition goes from resolution 1 to the given maximum resolution.

**Value**

Structure containing

original	original signal.
Wf	dyadic wavelet transform of signal.
Sf	multiresolution of signal.
maxresoln	number of decomposition scales.
np	size of signal.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[dwinverse](#), [mrecons](#), [ext](#).

---

noisywave

*Noisy Gravitational Wave*

---

**Description**

Noisy gravitational wave.

**Usage**

```
data(noisywave)
```

**Format**

A vector containing 8192 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**

Carmona, R. A., W. L. Hwang and B Torresani (1998) *Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S*, Academic Press, San Diego.

---

npl

*Prepare Graphics Environment*

---

**Description**

Splits the graphics device into prescribed number of windows.

**Usage**

```
npl(nbrow)
```

**Arguments**

nbrow            number of plots.

---

plotResult *Plot Dyadic Wavelet Transform Extrema*

---

### Description

Plot extrema of dyadic wavelet transform.

### Usage

```
plotResult(result, original, maxresoln, scale=FALSE, yaxtype="s")
```

### Arguments

result	result.
original	input signal.
maxresoln	number of decomposition scales.
scale	when set, the extrema at each scale is plotted with the same scale.
yaxtype	y axis type (see R manual).

### References

See discussions in the text of “Time-Frequency Analysis”.

### See Also

[plotwt](#), [epl](#), [wpl](#).

---

plotwt *Plot Dyadic Wavelet Transform*

---

### Description

Plot dyadic wavelet transform.

### Usage

```
plotwt(original, psi, phi, maxresoln, scale=FALSE, yaxtype="s")
```

### Arguments

original	input signal.
psi	dyadic wavelet transform.
phi	scaling function transform at last resolution.
maxresoln	number of decomposition scales.
scale	when set, the wavelet transform at each scale is plotted with the same scale.
yaxtype	axis type (see R manual).

**References**

See discussions in the text of “Time-Frequency Analysis”.

**See Also**

[plotResult](#), [epl](#), [wpl](#).

---

purwave

*Pure Gravitational Wave*

---

**Description**

Pure gravitational wave.

**Usage**

data (purwave)

**Format**

A vector containing 8192 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**

Carmona, R. A., W. L. Hwang and B Torresani (1998) *Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S*, Academic Press, San Diego.

---

regrec

*Reconstruction from a Ridge*

---

**Description**

Reconstructs signal from a “regularly sampled” ridge, in the wavelet case.

**Usage**

```
regrec(sinput, cwtinput, phi, compr, noct, nvoice, epsilon=0,
w0=2 * pi, fast=FALSE, plot=FALSE, para=0, hflag=FALSE,
check=FALSE, minnbnodes=2, real=FALSE)
```

**Arguments**

siginput	input signal.
cwtinput	wavelet transform, output of <code>cwt</code> .
phi	unsampled ridge.
compr	subsampling rate for the wavelet coefficients (at scale 1)
noct	number of octaves (powers of 2)
nvoice	number of different scales per octave
epsilon	coefficient of the $Q_2$ term in reconstruction kernel
w0	central frequency of Morlet wavelet
fast	if set to TRUE, the kernel is computed using trapezoidal rule.
plot	if set to TRUE, displays original and reconstructed signals
para	scale parameter for extrapolating the ridges.
hflag	if set to TRUE, uses $Q_1$ as first term in the kernel.
check	if set to TRUE, computes <code>cwt</code> of reconstructed signal.
minnbnodes	minimum number of nodes for the reconstruction.
real	if set to TRUE, uses only real constraints on the transform.

**Value**

Returns a list containing:

sol	reconstruction from a ridge.
A	<wavelets,dualwavelets> matrix.
lam	coefficients of dual wavelets in reconstructed signal.
dualwave	array containing the dual wavelets.
morvelets	array containing the wavelets on sampled ridge.
solskel	wavelet transform of sol, restricted to the ridge.
inputskel	wavelet transform of signal, restricted to the ridge.
Q2	second part of the reconstruction kernel.
nbnodes	number of nodes used for the reconstruction.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[regrec2](#), [ridrec](#), [gregrec](#), [gridrec](#).

regrec2

*Reconstruction from a Ridge***Description**

Reconstructs signal from a “regularly sampled” ridge, in the wavelet case, from a precomputed kernel.

**Usage**

```
regrec2(sinput, cwtinput, phi, nbnodes, noct, nvoice, Q2,
epsilon=0.5, w0=2 * pi, plot=FALSE)
```

**Arguments**

sinput	input signal.
cwtinput	wavelet transform, output of <code>cwt</code> .
phi	unsampled ridge.
nbnodes	number of samples on the ridge
noct	number of octaves (powers of 2)
nvoice	number of different scales per octave
Q2	second term of the reconstruction kernel
epsilon	coefficient of the $Q_2$ term in reconstruction kernel
w0	central frequency of Morlet wavelet
plot	if set to TRUE, displays original and reconstructed signals

**Details**

The computation of the kernel may be time consuming. This function avoids recomputing it if it was computed already.

**Value**

Returns a list containing:

sol	reconstruction from a ridge.
A	<wavelets,dualwavelets> matrix.
lam	coefficients of dual wavelets in reconstructed signal.
dualwave	array containing the dual wavelets.
morvelets	array containing the wavelets on sampled ridge.
solskel	wavelet transform of sol, restricted to the ridge.
inputskel	wavelet transform of signal, restricted to the ridge.
nbnodes	number of nodes used for the reconstruction.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[regrec](#), [gregrec](#), [ridrec](#), [sridrec](#).

---

RidgeSampling

*Sampling Gabor Ridge*

---

**Description**

Given a ridge  $\phi$  (for the Gabor transform), returns a (regularly) subsampled version of length `nbnodes`.

**Usage**

```
RidgeSampling(phi, nbnodes)
```

**Arguments**

<code>phi</code>	ridge (1D array).
<code>nbnodes</code>	number of samples.

**Details**

Gabor ridges are sampled uniformly.

**Value**

Returns a list containing the discrete values of the ridge.

<code>node</code>	time coordinates of the ridge samples.
<code>phinode</code>	frequency coordinates of the ridge samples.

**References**

See discussions in the text of "Time-Frequency Analysis".

**See Also**

[wRidgeSampling](#).

---

 ridrec *Reconstruction from a Ridge*


---

**Description**

Reconstructs signal from sample of a ridge, in the wavelet case.

**Usage**

```
ridrec(cwtinput, node, phinode, noct, nvoice, Qinv, epsilon, np,
w0=2 * pi, check=FALSE, real=FALSE)
```

**Arguments**

cwtinput	wavelet transform, output of <code>cwt</code> .
node	time coordinates of the ridge samples.
phinode	scale coordinates of the ridge samples.
noct	number of octaves (powers of 2).
nvoice	number of different scales per octave.
Qinv	inverse of the matrix $Q$ of the quadratic form.
epsilon	coefficient of the $Q_2$ term in reconstruction kernel
np	number of samples of the reconstructed signal.
w0	central frequency of Morlet wavelet.
check	if set to TRUE, computes <code>cwt</code> of reconstructed signal.
real	if set to TRUE, uses only constraints on the real part of the transform.

**Value**

Returns a list containing the reconstructed signal and the chained ridges.

sol	reconstruction from a ridge
A	<wavelets,dualwavelets> matrix
lam	coefficients of dual wavelets in reconstructed signal.
dualwave	array containing the dual wavelets.
morvelets	array of morlet wavelets located on the ridge samples.
solskel	wavelet transform of sol, restricted to the ridge
inputskel	wavelet transform of signal, restricted to the ridge

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[sridrec](#), [regrec](#), [regrec2](#).

---

`rkernel`*Kernel for Reconstruction from Wavelet Ridges*

---

### Description

Computes the cost from the sample of points on the estimated ridge and the matrix used in the reconstruction of the original signal, in the case of real constraints. Modification of the function [kernel](#).

### Usage

```
rkernel(node, phinode, nvoice, x.inc=1, x.min=node[1],  
x.max=node[length(node)], w0=2 * pi, plot=FALSE)
```

### Arguments

<code>node</code>	values of the variable $b$ for the nodes of the ridge.
<code>phinode</code>	values of the scale variable $a$ for the nodes of the ridge.
<code>nvoice</code>	number of scales within 1 octave.
<code>x.inc</code>	step unit for the computation of the kernel.
<code>x.min</code>	minimal value of $x$ for the computation of $Q_2$ .
<code>x.max</code>	maximal value of $x$ for the computation of $Q_2$ .
<code>w0</code>	central frequency of the wavelet.
<code>plot</code>	if set to TRUE, displays the modulus of the matrix of $Q_2$ .

### Details

Uses Romberg's method for computing the kernel.

### Value

matrix of the  $Q_2$  kernel

### References

See discussions in the text of "Time-Frequency Analysis".

### See Also

[kernel](#), [fastkernel](#), [gkernel](#), [zerokernel](#).

## Description

Reconstructs signal from ridges obtained by `crc`, using the restriction of the transform to the ridge.

## Usage

```
scrcrec(sinput, tinput, beemap, bstep=5, ptile=0.01, plot=2)
```

## Arguments

<code>sinput</code>	input signal.
<code>tinput</code>	time-frequency representation (output of <code>cwt</code> or <code>cgt</code> ).
<code>beemap</code>	output of crazy climber algorithm
<code>bstep</code>	used for the chaining (see <code>cfamily</code> ).
<code>ptile</code>	threshold on the measure beemap (see <code>cfamily</code> ).
<code>plot</code>	1: displays signal, components, and reconstruction one after another. 2: displays signal, components and reconstruction. Else, no plot.

## Value

Returns a list containing the reconstructed signal and the chained ridges.

<code>rec</code>	reconstructed signal
<code>ordered</code>	image of the ridges (with different colors)
<code>comp</code>	2D array containing the signals reconstructed from ridges

## References

See discussions in the text of “Practical Time-Frequency Analysis”.

## See Also

`crc`, `cfamily` for crazy climbers method, `scrcrec` for reconstruction methods.

---

 skeleton

*Reconstruction from Dual Wavelets*


---

### Description

Computes the reconstructed signal from the ridge, given the inverse of the matrix Q.

### Usage

```
skeleton(cwtinput, Qinv, morvelets, bridge, aridge, N)
```

### Arguments

cwtinput	continuous wavelet transform (as the output of cwt)
Qinv	inverse of the reconstruction kernel (2D array)
morvelets	array of Morlet wavelets located at the ridge samples
bridge	time coordinates of the ridge samples
aridge	scale coordinates of the ridge samples
N	size of reconstructed signal

### Value

Returns a list of the elements of the reconstruction of a signal from sample points of a ridge

sol	reconstruction from a ridge
A	matrix of the inner products
lam	coefficients of dual wavelets in reconstructed signal. They are the Lagrange multipliers $\lambda$ 's of the text.
dualwave	array containing the dual wavelets.

### References

See discussions in the text of “Practical Time-Frequency Analysis”.

### See Also

[skeleton2](#), [zeroskeleton](#), [zeroskeleton2](#).

---

`skeleton2`*Reconstruction from Dual Wavelet*

---

**Description**

Computes the reconstructed signal from the ridge in the case of real constraints.

**Usage**

```
skeleton2(cwtinput, Qinv, morvelets, bridge, aridge, N)
```

**Arguments**

<code>cwtinput</code>	continuous wavelet transform (as the output of <code>cwt</code> ).
<code>Qinv</code>	inverse of the reconstruction kernel (2D array).
<code>morvelets</code>	array of Morlet wavelets located at the ridge samples.
<code>bridge</code>	time coordinates of the ridge samples.
<code>aridge</code>	scale coordinates of the ridge samples.
<code>N</code>	size of reconstructed signal.

**Value**

Returns a list of the elements of the reconstruction of a signal from sample points of a ridge

<code>sol</code>	reconstruction from a ridge.
<code>A</code>	matrix of the inner products.
<code>lam</code>	coefficients of dual wavelets in reconstructed signal. They are the Lagrange multipliers $\lambda$ 's of the text.
<code>dualwave</code>	array containing the dual wavelets.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[skeleton.](#)

---

 smoothts

*Smoothing Time Series*


---

**Description**

Smooth a time series by averaging window.

**Usage**

```
smoothts(ts, windowsize)
```

**Arguments**

ts	Time series.
windowsize	Length of smoothing window.

**Value**

Smoothed time series (1D array).

**References**

See discussions in the text of “Time-Frequency Analysis”.

---

 smoothwt

*Smoothing and Time Frequency Representation*


---

**Description**

smooth the wavelet (or Gabor) transform in the time direction.

**Usage**

```
smoothwt(modulus, subrate, flag=FALSE)
```

**Arguments**

modulus	Time-Frequency representation (real valued).
subrate	Length of smoothing window.
flag	If set to TRUE, subsample the representation.

**Value**

2D array containing the smoothed transform.

## References

See discussions in the text of “Time-Frequency Analysis”.

## See Also

[corona](#), [coronoid](#), [snake](#), [snakoid](#).

---

 snake

*Ridge Estimation by Snake Method*


---

## Description

Estimate a ridge from a time-frequency representation, using the snake method.

## Usage

```
snake(tfrep, guessA, guessB, snakesize=length(guessB),
      tfspec=numeric(dim(modulus)[2]), subrate=1, temprate=3, muA=1,
      muB=muA, lambdaB=2 * muB, lambdaA=2 * muA, iteration=1000000,
      seed=-7, costsub=1, stagnant=20000, plot=TRUE)
```

## Arguments

<code>tfrep</code>	Time-Frequency representation (real valued).
<code>guessA</code>	Initial guess for the algorithm (frequency variable).
<code>guessB</code>	Initial guess for the algorithm (time variable).
<code>snakesize</code>	the length of the initial guess of time variable.
<code>tfspec</code>	Estimate for the contribution of the noise to modulus.
<code>subrate</code>	Subsampling rate for ridge estimation.
<code>temprate</code>	Initial value of temperature parameter.
<code>muA</code>	Coefficient of the ridge’s derivative in cost function (frequency component).
<code>muB</code>	Coefficient of the ridge’s derivative in cost function (time component).
<code>lambdaB</code>	Coefficient of the ridge’s second derivative in cost function (time component).
<code>lambdaA</code>	Coefficient of the ridge’s second derivative in cost function (frequency component).
<code>iteration</code>	Maximal number of moves.
<code>seed</code>	Initialization of random number generator.
<code>costsub</code>	Subsampling of cost function in output.
<code>stagnant</code>	maximum number of steps without move (for the stopping criterion)
<code>plot</code>	when set (by default), certain results will be displayed

**Value**

Returns a structure containing:

ridge	1D array (of same length as the signal) containing the ridge.
cost	1D array containing the cost function.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[corona](#), [coronoid](#), [icm](#), [snakoid](#).

---

snakeview

*Restriction to a Snake*


---

**Description**

Restrict time-frequency transform to a snake.

**Usage**

```
snakeview(modulus, snake)
```

**Arguments**

modulus	Time-Frequency representation (real valued).
snake	Time and frequency components of a snake.

**Details**

Recall that a snake is a (two components)  $\mathbb{R}$  structure.

**Value**

2D array containing the restriction of the transform modulus to the snake.

**References**

See discussions in the text of "Time-Frequency Analysis".

snakoid

*Modified Snake Method***Description**

Estimate a ridge from a time-frequency representation, using the modified snake method (modified cost function).

**Usage**

```
snakoid(modulus, guessA, guessB, snakesize=length(guessB),
        tfspec=numeric(dim(modulus)[2]), subrate=1, temprate=3, muA=1,
        muB=muA, lambdaB=2 * muB, lambdaA=2 * muA, iteration=1000000,
        seed=-7, costsub=1, stagnant=20000, plot=TRUE)
```

**Arguments**

modulus	Time-Frequency representation (real valued).
guessA	Initial guess for the algorithm (frequency variable).
guessB	Initial guess for the algorithm (time variable).
snakesize	The length of the first guess of time variable.
tfspec	Estimate for the contribution of srthe noise to modulus.
subrate	Subsampling rate for ridge estimation.
temprate	Initial value of temperature parameter.
muA	Coefficient of the ridge's derivative in cost function (frequency component).
muB	Coefficient of the ridge's derivative in cost function (time component).
lambdaB	Coefficient of the ridge's second derivative in cost function (time component).
lambdaA	Coefficient of the ridge's second derivative in cost function (frequency component).
iteration	Maximal number of moves.
seed	Initialization of random number generator.
costsub	Subsampling of cost function in output.
stagnant	Maximum number of stationary iterations before stopping.
plot	when set(default), some results will be displayed

**Value**

Returns a structure containing:

ridge	1D array (of same length as the signal) containing the ridge.
cost	1D array containing the cost function.
plot	when set(default), some results will be displayed.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[corona](#), [coronoid](#), [icm](#), [snake](#).

---

sridrec

*Simple Reconstruction from Ridge*

---

**Description**

Simple reconstruction of a real valued signal from a ridge, by restriction of the transform to the ridge.

**Usage**

```
sridrec(tfinput, ridge)
```

**Arguments**

tfinput	time-frequency representation.
ridge	ridge (1D array).

**Value**

(real) reconstructed signal (1D array)

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[ridrec](#), [gridrec](#).

---

SVD

*Singular Value Decomposition*

---

**Description**

Computes singular value decomposition of a matrix.

**Usage**

```
SVD(a)
```

**Arguments**

a                   input matrix.

**Details**

R interface for Numerical Recipes singular value decomposition routine.

**Value**

a structure containing the 3 matrices of the singular value decomposition of the input.

**References**

See discussions in the text of "Time-Frequency Analysis".

---

tfgmax

*Time-Frequency Transform Global Maxima*

---

**Description**

Computes the maxima (for each fixed value of the time variable) of the modulus of a continuous wavelet transform.

**Usage**

```
tfgmax(input, plot=TRUE)
```

**Arguments**

input                wavelet transform (as the output of the function `cwt`)  
plot                 if set to TRUE, displays the values of the energy as a function of the scale.

**Value**

output	values of the maxima (1D array)
pos	positions of the maxima (1D array)

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[tflmax](#).

---

tflmax

*Time-Frequency Transform Local Maxima*

---

**Description**

Computes the local maxima (for each fixed value of the time variable) of the modulus of a time-frequency transform.

**Usage**

```
tflmax(input, plot=TRUE)
```

**Arguments**

input	time-frequency transform (real 2D array).
plot	if set to T, displays the local maxima on the graphic device.

**Value**

values of the maxima (2D array).

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[tfgmax](#).

---

tfmean	<i>Average frequency by frequency</i>
--------	---------------------------------------

---

**Description**

Compute the mean of time-frequency representation frequency by frequency.

**Usage**

```
tfmean(input, plot=TRUE)
```

**Arguments**

input	time-frequency transform (output of <a href="#">cwt</a> or <a href="#">cgt</a> ).
plot	if set to T, displays the values of the energy as a function of the scale (or frequency).

**Value**

1D array containing the noise estimate.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[tfpct](#), [tfvar](#).

---

tfpct	<i>Percentile frequency by frequency</i>
-------	--

---

**Description**

Compute a percentile of time-frequency representation frequency by frequency.

**Usage**

```
tfpct(input, percent=0.8, plot=TRUE)
```

**Arguments**

input	time-frequency transform (output of <a href="#">cwt</a> or <a href="#">cgt</a> ).
percent	percentile to be retained.
plot	if set to T, displays the values of the energy as a function of the scale (or frequency).

**Value**

1D array containing the noise estimate.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[tfmean](#), [tfvar](#).

---

tfvar	<i>Variance frequency by frequency</i>
-------	--

---

**Description**

Compute the variance of time-frequency representation frequency by frequency.

**Usage**

```
tfvar(input, plot=TRUE)
```

**Arguments**

input	time-frequency transform (output of <a href="#">cwt</a> or <a href="#">cgt</a> ).
plot	if set to T, displays the values of the energy as a function of the scale (or frequency).

**Value**

1D array containing the noise estimate.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[tfmean](#), [tfpct](#).

---

Undocumented

*Undocumented Functions in Rwave*

---

**Description**

Numerous functions were not documented in the original Swave help files.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

---

vDOG

*DOG Wavelet Transform on one Voice*

---

**Description**

Compute DOG wavelet transform at one scale.

**Usage**

```
vDOG(input, scale, moments)
```

**Arguments**

<code>input</code>	Input signal (1D array).
<code>scale</code>	Scale at which the wavelet transform is to be computed.
<code>moments</code>	number of vanishing moments.

**Value**

1D (complex) array containing wavelet transform at one scale.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[vgt](#), [vwt](#).

---

vecgabor *Gabor Functions on a Ridge*

---

### Description

Generate Gabor functions at specified positions on a ridge.

### Usage

```
vecgabor(sigsize, nbnodes, location, frequency, scale)
```

### Arguments

sigsize	Signal size.
nbnodes	Number of wavelets to be generated.
location	b coordinates of the ridge samples (1D array of length nbnodes).
frequency	frequency coordinates of the ridge samples (1D array of length nbnodes).
scale	size parameter for the Gabor functions.

### Value

size parameter for the Gabor functions.

### See Also

[vecmorlet](#).

---

vecmorlet *Morlet Wavelets on a Ridge*

---

### Description

Generate Morlet wavelets at specified positions on a ridge.

### Usage

```
vecmorlet(sigsize, nbnodes, bridge, aridge, w0=2 * pi)
```

### Arguments

sigsize	Signal size.
nbnodes	Number of wavelets to be generated.
bridge	b coordinates of the ridge samples (1D array of length nbnodes).
aridge	a coordinates of the ridge samples (1D array of length nbnodes).
w0	Center frequency of the wavelet.

**Value**

2D (complex) array containing wavelets located at the specific points.

**See Also**

[vecgabor](#).

---

vgt

*Gabor Transform on one Voice*

---

**Description**

Compute Gabor transform for fixed frequency.

**Usage**

```
vgt(input, frequency, scale, plot=FALSE)
```

**Arguments**

<code>input</code>	Input signal (1D array).
<code>frequency</code>	frequency at which the Gabor transform is to be computed.
<code>scale</code>	frequency at which the Gabor transform is to be computed.
<code>plot</code>	if set to TRUE, plots the real part of cgt on the graphic device.

**Value**

1D (complex) array containing Gabor transform at specified frequency.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[vwt](#), [vDOG](#).

---

vwt *Voice Wavelet Transform*

---

**Description**

Compute Morlet's wavelet transform at one scale.

**Usage**

```
vwt(input, scale, w0=2 * pi)
```

**Arguments**

input	Input signal (1D array).
scale	Scale at which the wavelet transform is to be computed.
w0	Center frequency of the wavelet.

**Value**

1D (complex) array containing wavelet transform at one scale.

**References**

See discussions in the text of "Practical Time-Frequency Analysis".

**See Also**

[vgt](#), [vDOG](#).

---

wpl *Plot Dyadic Wavelet Transform.*

---

**Description**

Plot dyadic wavelet transform(output of [mw](#)).

**Usage**

```
wpl(dwtrans)
```

**Arguments**

dwtrans	dyadic wavelet transform (output of <a href="#">mw</a> ).
---------	---

**See Also**

[mw](#), [ext](#), [epl](#).

---

wRidgeSampling	<i>Sampling wavelet Ridge</i>
----------------	-------------------------------

---

**Description**

Given a ridge  $\phi$  (for the wavelet transform), returns a (appropriately) subsampled version with a given subsampling rate.

**Usage**

```
wRidgeSampling(phi, compr, nvoice)
```

**Arguments**

phi	ridge (1D array).
compr	subsampling rate for the ridge.
nvoice	number of voices per octave.

**Details**

To account for the variable sizes of wavelets, the sampling rate of a wavelet ridge is not uniform, and is proportional to the scale.

**Value**

Returns a list containing the discrete values of the ridge.

node	time coordinates of the ridge samples.
phinode	scale coordinates of the ridge samples.
nbnode	number of nodes of the ridge samples.

**See Also**

[RidgeSampling](#).

---

`wspec.pl`*Log of Wavelet Spectrum Plot*

---

**Description**

Displays normalized log of wavelet spectrum.

**Usage**

```
wspec.pl(wspec, nvoice)
```

**Arguments**

<code>wspec</code>	wavelet spectrum.
<code>nvoice</code>	number of voices.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[hurst.est.](#)

---

`WV`*Wigner-Ville function*

---

**Description**

Compute the Wigner-Ville transform, without any smoothing.

**Usage**

```
WV(input, nvoice, freqstep = (1/nvoice), plot = TRUE)
```

**Arguments**

<code>input</code>	input signal (possibly complex-valued)
<code>nvoice</code>	number of frequency bands
<code>freqstep</code>	sampling rate for the frequency axis
<code>plot</code>	if set to TRUE, displays the modulus of CWT on the graphic device.

**Value**

(complex) Wigner-Ville transform.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

---

YN

*Logarithms of the Prices of Japanese Yen*

---

**Description**

Logarithms of the prices of a contract of Japanese yen.

**Usage**

`data(YN)`

**Format**

A vector containing 500 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

**References**

Carmona, R. A., W. L. Hwang and B Torresani (1998) *Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S*, Academic Press, San Diego.

---

YNdiff

*Daily differences of Japanese Yen*

---

**Description**

Daily differences of [YN](#).

**Usage**

`data(YNdiff)`

**Format**

A vector containing 499 observations.

**Source**

See discussions in the text of “Practical Time-Frequency Analysis”.

## References

Carmona, R. A., W. L. Hwang and B Torresani (1998) *Practical Time-Frequency Analysis: Gabor and Wavelet Transforms with an Implementation in S*, Academic Press, San Diego.

---

zerokernel	<i>Reconstruction from Wavelet Ridges</i>
------------	---

---

## Description

Generate a zero kernel for reconstruction from ridges.

## Usage

```
zerokernel(x.inc=1, x.min, x.max)
```

## Arguments

x.min	minimal value of x for the computation of $Q_2$ .
x.max	maximal value of x for the computation of $Q_2$ .
x.inc	step unit for the computation of the kernel.

## Value

matrix of the  $Q_2$  kernel

## See Also

[kernel](#), [fastkernel](#), [gkernel](#), [gkernel](#).

---

zeroskeleton	<i>Reconstruction from Dual Wavelets</i>
--------------	--

---

## Description

Computes the the reconstructed signal from the ridge when the epsilon parameter is set to zero

## Usage

```
zeroskeleton(cwtinput, Qinvs, morvelets, bridge, aridge, N)
```

**Arguments**

<code>cwtinput</code>	continuous wavelet transform (as the output of <code>cwt</code> ).
<code>Qinv</code>	inverse of the reconstruction kernel (2D array).
<code>morvelets</code>	array of Morlet wavelets located at the ridge samples.
<code>bridge</code>	time coordinates of the ridge samples.
<code>aridge</code>	scale coordinates of the ridge samples.
<code>N</code>	size of reconstructed signal.

**Details**

The details of this reconstruction are the same as for the function `skeleton`. They can be found in the text

**Value**

Returns a list of the elements of the reconstruction of a signal from sample points of a ridge

<code>sol</code>	reconstruction from a ridge.
<code>A</code>	matrix of the inner products.
<code>lam</code>	coefficients of dual wavelets in reconstructed signal. They are the Lagrange multipliers $\lambda$ 's of the text.
<code>dualwave</code>	array containing the dual wavelets.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[skeleton](#), [skeleton2](#), [zeroskeleton2](#).

---

`zeroskeleton2`

*Reconstruction from Dual Wavelets*

---

**Description**

Computes the the reconstructed signal from the ridge when the epsilon parameter is set to zero, in the case of real constraints.

**Usage**

```
zeroskeleton2(cwtinput, Qinv, morvelets, bridge, aridge, N)
```

**Arguments**

<code>cwtinput</code>	continuous wavelet transform (output of <code>cwt</code> ).
<code>Qinv</code>	inverse of the reconstruction kernel (2D array).
<code>morvelets</code>	array of Morlet wavelets located at the ridge samples.
<code>bridge</code>	time coordinates of the ridge samples.
<code>aridge</code>	scale coordinates of the ridge samples.
<code>N</code>	size of reconstructed signal.

**Details**

The details of this reconstruction are the same as for the function `skeleton`. They can be found in the text

**Value**

Returns a list of the elements of the reconstruction of a signal from sample points of a ridge

<code>sol</code>	reconstruction from a ridge.
<code>A</code>	matrix of the inner products.
<code>lam</code>	coefficients of dual wavelets in reconstructed signal. They are the Lagrange multipliers $\lambda$ 's of the text.
<code>dualwave</code>	array containing the dual wavelets.

**References**

See discussions in the text of “Practical Time-Frequency Analysis”.

**See Also**

[skeleton](#), [skeleton2](#), [zeroskeleton](#).

# Index

## \*Topic **datasets**

- A0, 2
- A4, 2
- amber7, 3
- amber8, 4
- amber9, 4
- B0, 5
- B4, 5
- back1.000, 6
- back1.180, 6
- back1.220, 7
- C0, 7
- C4, 8
- ch, 10
- click, 12
- D0, 23
- D4, 23
- Ekg, 25
- HOWAREYOU, 37
- noisywave, 47
- purwave, 49
- YN, 73
- YNdiff, 73

## \*Topic **hplot**

- wspec.pl, 72

## \*Topic **ts**

- adjust.length, 3
- cfamily, 8
- cgt, 9
- check.maxresoln, 11
- cleanph, 11
- corona, 12
- coronoid, 13
- crc, 15
- crcrec, 16
- crfview, 17
- cwt, 17
- cwtimage, 18
- cwtp, 19

- cwtpolar, 20
- cwtsquiz, 21
- cwtTh, 22
- DOG, 24
- dwinverse, 25
- epl, 26
- ext, 26
- fastgkernel, 27
- fastkernel, 28
- gabor, 30
- gcrcrec, 31
- gkernel, 32
- gregrec, 33
- gridrec, 34
- gsampleOne, 35
- gwave, 36
- gwave2, 36
- hurst.est, 38
- icm, 39
- kernel, 40
- mbtrim, 41
- mntrim, 42
- morlet, 43
- morwave, 44
- morwave2, 44
- mrecons, 45
- mw, 46
- npl, 47
- plotResult, 48
- plotwt, 48
- regrec, 49
- regrec2, 51
- RidgeSampling, 52
- ridrec, 53
- rkernel, 54
- scrcrec, 55
- skeleton, 56
- skeleton2, 57
- smoothts, 58

- smoothwt, 58
  - snake, 59
  - snakeview, 60
  - snakoid, 61
  - sridrec, 62
  - SVD, 63
  - tfgmax, 63
  - tflmax, 64
  - tfmean, 65
  - tfpct, 65
  - tfvar, 66
  - Undocumented, 67
  - vDOG, 67
  - vecgabor, 68
  - vecmorlet, 68
  - vgt, 69
  - vwt, 70
  - wpl, 70
  - wRidgeSampling, 71
  - wspec.pl, 72
  - WV, 72
  - zerokernel, 74
  - zeroskeleton, 74
  - zeroskeleton2, 75
- A0, 2
  - A4, 2
  - adjust.length, 3
  - amber7, 3
  - amber8, 4
  - amber9, 4
- B0, 5
  - B4, 5
  - back1.000, 6
  - back1.180, 6
  - back1.220, 7
  - band (Undocumented), 67
- C0, 7
  - C4, 8
  - cfamily, 8, 9, 15, 17, 32, 55
  - cgt, 9, 20, 21, 24, 33, 34, 55, 65, 66
  - cgtRadar (Undocumented), 67
  - ch, 10
  - check.maxresoln, 11
  - cleanph, 11
  - click, 12
  - cloudXYZ (Undocumented), 67
  - confident (Undocumented), 67
  - corona, 12, 13–15, 39, 59, 60, 62
  - coronoid, 13, 13–15, 39, 59, 60, 62
  - crc, 8, 9, 15, 16, 17, 31, 32, 55
  - crcirgrec (Undocumented), 67
  - crcirrec (Undocumented), 67
  - crcrec, 9, 15, 16, 32, 55
  - crfview, 17
  - cwt, 10, 17, 19–22, 24, 33, 50, 51, 53, 55, 63, 65, 66, 75, 76
  - cwtimage, 18, 21
  - cwtp, 10, 18, 19, 21, 22, 24
  - cwtpolar, 19, 20
  - cwtsquiz, 10, 21, 24
  - cwtTh, 18, 20, 22
- D0, 23
  - D4, 23
  - ddw (Undocumented), 67
  - DOG, 10, 18–22, 24
  - dw (Undocumented), 67
  - dwinverse, 25, 47
- Ekg, 25
  - epl, 26, 48, 49, 70
  - ext, 25, 26, 26, 41, 42, 46, 47, 70
- fastgkernel, 27, 33
  - fastkernel, 28, 28, 33, 54, 74
  - fftshift (Undocumented), 67
- gabor, 18, 20, 22, 30, 43
  - gcrrec, 9, 15, 31
  - girregrec (Undocumented), 67
  - gkernel, 28, 29, 32, 35, 40, 54, 74
  - gregrec, 33, 35, 50, 52
  - gridrec, 34, 50, 62
  - gsampleOne, 35
  - gwave, 36, 36, 37, 44, 45
  - gwave2, 36, 36, 44, 45
- hesrcrc (Undocumented), 67
  - HOWAREYOU, 37
  - hurst.est, 38, 72
- icm, 13–15, 39, 39, 60, 62
  - irregrec (Undocumented), 67
- kernel, 29, 33, 35, 40, 54, 74

- mbpval (*Undocumented*), 67
- mbtrim, 41, 42
- mcgt (*Undocumented*), 67
- mpval (*Undocumented*), 67
- mtrim, 41, 42
- morlet, 30, 43
- morwave, 36, 37, 44, 45
- morwave2, 36, 37, 44, 44
- mrecons, 11, 25, 27, 41, 42, 45, 47
- mw, 11, 25–27, 46, 46, 70
  
- noisywave, 47
- npl, 47
  
- pcacrc (*Undocumented*), 67
- pcafamily (*Undocumented*), 67
- pcamaxima (*Undocumented*), 67
- pcamorwave (*Undocumented*), 67
- pcarec (*Undocumented*), 67
- pcaregrec (*Undocumented*), 67
- PcaRidgeSampling (*Undocumented*), 67
- pccaridrec (*Undocumented*), 67
- pczeroskeleton (*Undocumented*), 67
- plotResult, 48, 49
- plotwt, 48, 48
- purwave, 49
  
- regrec, 34, 35, 49, 52, 53
- regrec2, 35, 50, 51, 53
- RidgeDist (*Undocumented*), 67
- RidgeIrregSampling (*Undocumented*), 67
- RidgeSampling, 52, 71
- ridrec, 50, 52, 53, 62
- rkernl, 28, 29, 33, 40, 54
- robustrec (*Undocumented*), 67
- RunRec (*Undocumented*), 67
  
- SampleGen (*Undocumented*), 67
- Sausage (*Undocumented*), 67
- scrcrc, 9, 15, 17, 32, 55
- showRadar (*Undocumented*), 67
- simplepcarec (*Undocumented*), 67
- skeleton, 56, 57, 75, 76
- skeleton2, 56, 57, 75, 76
- smoothts, 58
- smoothwt, 58
- snake, 13–15, 39, 59, 59, 62
- snakeview, 60
- snakoid, 13–15, 39, 59, 60, 61
- SpecGen (*Undocumented*), 67
- sridrec, 35, 52, 53, 62
- SVD, 63
  
- tfgmax, 63, 64
- tflmax, 64, 64
- tfmean, 38, 65, 66
- tfpct, 65, 65, 66
- tfvar, 65, 66, 66
  
- Undocumented, 67
  
- vDOG, 67, 69, 70
- vecgabor, 68, 69
- vecmorlet, 68, 68
- vgt, 67, 69, 70
- vwt, 67, 69, 70
- vwtTh (*Undocumented*), 67
  
- wpl, 26, 48, 49, 70
- wRidgeSampling, 52, 71
- wspec.pl, 38, 72
- WV, 72
  
- YN, 73, 73
- YNdiff, 73
  
- zerokernel, 28, 29, 33, 40, 54, 74
- zeroskeleton, 56, 74, 76
- zeroskeleton2, 56, 75, 75