The sound Package

April 25, 2008

Version 1.2

Date 2008-04-22

Title A Sound Interface for R

Author Matthias Heymann <mail@MatthiasHeymann.de>

Maintainer Matthias Heymann <mail@MatthiasHeymann.de>

Depends R (>= 1.4.1)

Description Basic functions for dealing with wav files and sound samples.

License GPL version 2 or newer

URL http://www.MatthiasHeymann.de

R topics documented:

`Ops.Sample` *Basic Operations for Sample Objects*

Description

These functions apply the basic operations pointwise to the waveforms of the samples.

Usage

```
e1 + e2
e1 - e2
e1 * e2
const * s
s * const
s / const
sum.Sample(e1, e2, ...)
prod.Sample(e1, e2, ...)
```

Arguments

`s, e1, e2, ...`	For `+`,`-`,`*`,`/`, Sample objects. For `prod.Sample` and `sum.Sample`, Sample objects or strings giving the names of wav files.
`const`	a double.

Details

The sum of two Sample objects corresponds to the sound when both samples are played at a time. The product of two samples causes an effect called ringmodulation, but it can also be used to add some vibrato to a sound (see the examples).

If the samples have different sample parameters (bits, rate and channels), the system uses the command `fitSampleParameters` to adjust them before the waveforms are combined.

Be careful to make sure that the resulting waveform does not exceed the interval [-1,1] when it is played or saved to disk, otherwise you will lose information and hear cracks in the sound. To avoid this, you can use `const * s` or the `normalize` command.

In `prod` and `sum` also the names of wavefiles can be used. Other forms like `e1+e2` do not accept filenames as arguments. If the first argument `e1` is a filename, the explicit forms `sum.Sample` and `prod.Sample` must be used.

Value

a Sample object.

Author(s)

Matthias Heymann

See Also

`normalize`, `center`

Examples

```
## Not run:
e1 <- Sine(440,1)
e2 <- Sine(220,1)
play((e1+e2)/2) # both samples at a time

play(Sine(440,1)*Sine(5,1)) # vibrato
## End(Not run)
```

Sample *Sample Objects*

Description

`as.Sample` creates a Sample object from a given numeric matrix.

`is.Sample` tests if its argument is a Sample object or the name of a wav file.

Usage

```
as.Sample(sound, rate, bits)
is.Sample(s, argname="'s' ")
```

Arguments

sound	a channels(s) x sampleLength(s) matrix or a vector of doubles describing the waveform(s) of the sample.
rate	the sampling rate (number of samples per second).
bits	the sampling quality (the number of bits per sample), 8 or 16.
s	an R object to be tested.
argname	a string giving the name of the object that is tested. It is used for creating an error message.

Details

The rows of the matrix represent the channels of the sample: If sound is a vector or a matrix with only one row, as.Sample will return a mono sample; if sound is a matrix with two rows, as.Sample returns a stereo sample, where the left and the right channel are represented by the first and the second row, respectively.

sound can contain any real number, but when the Sample object is played or saved to disk, [-1,1] is regarded as the range of the sample, and any values outside this interval will cause cracks in the sound.

A Sample object's waveform can exceed this interval during calculations. It is the task of the programmer to take care of the range of the waveform before saving or playing the sample, for example by using the normalize command.

Internally, the sound is saved as a matrix with doubles, independent of the bits parameter that is only used when the Sample object is played or saved to disk.

The is.Sample command is used by several other routines that allow both Sample objects and filenames as arguments.

Value

For as.Sample a Sample object, that is a list with the components $sound, $rate and $bits.

is.Sample returns a list with the entries

test	a logical indicating whether or not s is a Sample object or the name of a valid wav file.
error	a string with one of the messages "Filename must have the extension .wav.", "File not found.", "No read permission for this file.", or "Argument "+ argname + "must be a Sample object or the name of a wav file." If test=TRUE, this list entry doesn't exist.

Author(s)

Matthias Heymann

See Also

`stereo` for creating a stereo Sample object from two mono Sample objects, `loadSample` for loading a wav file and turning it into a Sample object, `saveSample` for saving a Sample object as a wav file, `sound`, `bits`, `rate`, `channels`, `sampleLength` and `duration` for access to the basic parameters of a Sample object.

Examples

```
## Not run:
waveLeft <- 2*((seq(0,80,length=88200)%%1^2)-.5)
s <- as.Sample(waveLeft,44100,16)
play(s) # a mono sample

waveRight <- waveLeft[88200:1]
s <- as.Sample(rbind(waveLeft,waveRight),44100,16)
play(s) # a stereo Sample

# How to use is.Sample to allow both a Sample object and a filename
# as an argument:
x <- anyargument
sampletest <- is.Sample(x, argname="'x' ")
if (!sampletest$test) stop(sampletest$error) #no valid argument
x <- loadSample(x,filecheck=FALSE)
# If x is Sample object, loadSample will return it immediately.
# If x is a string, the Sample object will be loaded from disk.
# No check for existence of the file will be performed since this
# was already tested in is.Sample.
#
# Now x is a Sample object, continue with code.
## End(Not run)
```

`Sine` *Create Sample Objects for the Basic waveforms*

Description

Create a Sample object with a sine, sawtooth, or square waveform, silence, or noise.

Usage

```
Sine(freq, dur, rate=44100, bits=16, channels=1)
Sawtooth(freq, dur, rate=44100, bits=16, channels=1, reverse=FALSE)
Square(freq, dur, rate=44100, bits=16, channels=1, upPerc=50)
Silence(dur, rate=8000, bits=8, channels=1)
Noise(dur, rate=44100, bits=16, channels=1)
```

Arguments

freq	the frequency (a double).
dur	the duration in seconds (a double).
rate	the sampling rate, an integer between 1000 and 48000.
bits	the sampling quality in bits per sample, 8 or 16.
channels	1 for mono, or 2 for stereo.
reverse	logical. If TRUE, the waveform will be mirrored vertically.
upPerc	a number between 0 and 100 giving the percentage of the waveform with value +1.

Details

If channels=2, left and right side of the sample will be the same for Sine, Sawtooth and Square. For Noise, both sides will be generated separately, using runif.

Value

a Sample object.

Author(s)

Matthias Heymann

See Also

as.Sample, loadSample, nullSample

Examples

```
## Not run:
s1 <- Sine(440,1)
play(s1)

s2 <- Sawtooth(440,1)
play(s2)

play(Noise(1))
## End(Not run)
```

`WavPlayer` *Set or Get the System Command for Playing WAV Files*

Description

`findWavPlayer` returns the most common system commands on your OS for playing wav files.

`WavPlayer` returns the command that is currently used by `play`.

`setWavPlayer` is used to define the command to be used by `play`.

Usage

```
findWavPlayer()
WavPlayer()
setWavPlayer(command=NULL)
```

Arguments

`command`	a vector of character strings giving the command to be used as `"command wavfile.wav"`. If it contains more than one string, the commands are tested one after the other, and the first one that works properly will be used for future calls of the `play` command. If `command=NULL`,the command `findWavPlayer()` is used to determine the standard commands for your system.

Details

The `play` command makes a system call of the form `"command wavfile.wav"`, where 'command' is the string returned by `WavPlayer()`.

The default commands are `'mplay32 /play'` (calling the Windows media player) for Win32-systems and `'play'` and `'playwave'` for Linux systems. Other commands will be added in future versions of this package.

While the Windows Media player is included in the standard Windows installation, playwave might have to be installed manually. Under RedHat Linux playwave is part of the SDL_mixer package. To download it, go to `http://www.libsdl.org/projects/SDL_mixer`.

But any other program that provides a system call of the above form to play wav files is also fine. Please report additional play commands to the author (send an email to ⟨mail@MatthiasHeymann.de⟩) so that they can be recognized automatically in future versions of this package.

`setWavPlayer` is called directly after loading the package.

When `setWavPlayer` is called, it tries to play an empty wav file, using the new command(s). If it fails, no changes are made.

Value

`WavPlayer` returns the wav play command that is currently used, or `NULL`, if none is selected yet.

`findWavPlayer` returns the default commands for your system, or `NULL`, if no command is known for your system.

Author(s)

Matthias Heymann

See Also

play for playing Sample objects or wav files.

Examples

```
## Not run:
setWavPlayer("playwave")
# tries to set the command "playwave wavfile.wav" as the
# preference for playing wav files with the play command.
# If successful,
WavPlayer()
# returns the string "playwave" afterwards.
## End(Not run)
```

---

appendSample *Append Sample Objects*

Description

Append two or more Sample objects or wav files.

Usage

```
appendSample(s1, s2, ...)
```

Arguments

s1, s2, ...	Sample objects, or the names of wav files.

Details

If the samples have different sample parameters (bits, rate and channels), the command fitSampleParameters is called to adjust them before the samples are appended.

Value

a Sample object with the samples played one after the other.

Author(s)

Matthias Heymann

See Also

cutSampleEnds to avoid cracks between two appended samples,

sum.Sample for playing several samples at a time.

Examples

```
## Not run:
s1 <- Sine(440,1)
s2 <- Sine(550,1)
s3 <- Sine(660,1)
s4 <- Sine(880,1)
play(appendSample(s1,s2,s3,s4))
## End(Not run)
```

---

bits *Bits per Sample*

Description

Get or set the bits parameter (the sampling quality) of a Sample object or a wav file.

Usage

```
bits(s)
bits(s) <- value
setBits(s,value)
```

Arguments

s	a Sample object, or a string giving the name of a wav file.
value	the number of bits per sample: 8, 16 or 24.

Details

The replacement form can be used to reset the sampling quality of a Sample object, that is the number of bits per sample (8 or 16). Here, filenames are not accepted.

Value

For bits, the bits parameter (number of bits per sample) of the Sample object (8, 16 or 24).

For setBits, a Sample object with the new bits parameter.

Note

Changing the sampling quality of a Sample object does not affect its actual data but only its `$bits` flag. The sampling quality is only used when a Sample object is played or saved to disk. Internally, R always uses doubles for the waveform.

An 8 bit sample needs only half the disk space compared to a 16 bit sample, but it has a lower sound quality.

Note also that 24 bit samples cannot be played by every wav file player.

Author(s)

Matthias Heymann

See Also

`fitSampleParameters`

Examples

```
## Not run:
s <- Sine(20000,1,rate=44100,bits=16)
play(s)
print(s)
bits(s) <- 8
play(s)  # now worse quality
print(s) # but less disk space
play(setBits(s,16)) # now better quality again, since waveform data was not changed.
## End(Not run)
```

---

center *Center a Sample Object.*

Description

This function adds a constant to a Sample object's waveform, so that its mean gets zero. This makes sense especially for sonification purposes, when (in general non-centered) data is transformed into sound.

Usage

```
center(s)
```

Arguments

`s`	a Sample object, or a string giving the name of a wav file.

Details

For a stereo Sample object, both channels are treated separately.

Value

a Sample object with zero as the mean of each channel's waveform.

Author(s)

Matthias Heymann

See Also

normalize

Examples

```
## Not run:
x <- seq(0,50*pi,length=10000)
waveform <- (sin(x))^2 + .6*cos(x/2)^2
s <- as.Sample(waveform,44100,16)
plot(s) # nice idea, but wrong range for a sample
play(s) # sounds ugly, too
s <- center(s)
plot(s)  # now zero is the mean
play(s)  # sounds good, but too quiet
s <- normalize(s)
plot(s)  # this looks like a perfect sample!
play(s)  # e voila!
## End(Not run)
```

---

channels *Number of Channels of a Sample Object*

Description

Get or set the number of channels of a Sample object or a wav file.

Usage

```
channels(s)
channels(s) <- value
setChannels(s,value)
```

Arguments

s	a Sample object, or a string giving the name of a wav file.
value	1 for mono, or 2 for stereo.

Details

The replacement form can be used to reset the number of channels of a Sample object (here, filenames are not accepted).

If a mono sample is transformed into a stereo sample, each channel of the stereo sample equals the waveform of the mono sample. If a stereo Sample is transformed to a mono sample, `(left(s)+right(s))/2` is returned.

Value

For `channels`, the number of channels of the sample (1 for mono, 2 for stereo).

For `setChannels`, a Sample object with the new `channels` parameter.

Author(s)

Matthias Heymann

See Also

`fitSampleParameters`

Examples

```
## Not run:
s <- stereo(Sine(440,1),Sine(220,1))
channels(s)  # 2
play(s)
channels(s) <- 1  # now a mono sample
play(s)
## End(Not run)
```

---

cutSample *Cut Sample Objects*

Description

Cut a part out of a Sample object.

Usage

```
cutSample(s, start, end)
## S3 method for class 'Sample':
s[i]
```

Arguments

s	a Sample object, or a string giving the name of a wav file.
start	the start position in seconds.
end	the end position in seconds.
i	a vector of integers giving the numbers of the columns in the waveform matrix to be used.

Details

Only the intersection of [start,end] with [0,duration(s)] is returned. Similarly, in the second form the intersection of v with 1:sampleLength(s) is returned.

Value

the specified part of the given sample as a new Sample object.

Author(s)

Matthias Heymann

See Also

sound for direct access to the waveform matrix,

cutSampleEnds and noSilence for special cutoff techniques.

Examples

```
## Not run:
s <- appendSample(Sine(330,1),Sine(440,1))
play(cutSample(s,.8,1.8))
play(s[(44100*.8):(44100*1.8)])  # the same
## End(Not run)
```

---

cutSampleEnds *Prepare Sample Object for appendSample*

Description

Prepare a Sample object or a wav file for usage of appendSample to avoid cracks between two appended samples.

Usage

```
cutSampleEnds(s)
```

Arguments

s	a Sample object, or a string giving the name of a wav file.

Details

At the beginning of the sample, all values in the waveform until the first transition from negative to positive values are dropped, at the end everything after the last transition from negative to positive values is dropped.

Currently, only channel 1 is used to determine which parts to drop. Hence in stereo samples there can still be some cracks in the right channel.

Value

a Sample object.

Author(s)

Matthias Heymann

See Also

cutSample, appendSample

Examples

```
## Not run:
s1 <- Sine(440,.01)
s2 <- Sine(550,.01)
s3 <- Sine(660,.01)
s4 <- Sine(880,.01)
l <- list(s1,s2,s3,s4)
#  first without cutSampleEnds:
s <- nullSample()
for (i in 1:99) {
  s <- appendSample(s,l[[i%%4+1]])
}
play(s) # ugly cracks
# now with cutSampleEnds:
s <- nullSample()
for (i in 1:99) {
  s <- appendSample(s,cutSampleEnds(l[[i%%4+1]]))
}
play(s) # no cracks,

# This is how it works:
# The waveform is not smooth between s1 and s2:
plot(appendSample(s1,s2))
# This is because s1 just ends somewhere at y=0.6:
plot(s1)
# Let's cut off the last positive part of it:
plot(cutSampleEnds(s1))
```

```
# A similar cuttoff would be made at the beginning
# of the sample (if it was necessary).
# Now the two samples fit perfectly (the cut is at x=400):
plot(appendSample(cutSampleEnds(s1),cutSampleEnds(s2)))
## End(Not run)
```

`duration` *Duration of a Sample Object*

Description

Get or set the duration (in seconds) of a Sample object or a wav file.

Usage

```
duration(s)
duration(s) <- value
setDuration(s,value)
```

Arguments

`s`	a Sample object, or a string giving the name of a wav file.
`value`	a double giving the duration in seconds.

Details

The replacement form can be used to reset the duration of the Sample object (here, filenames are not accepted).

If a Sample object is shortened, extra values are discarded. When a Sample object is lengthened, it is padded out to its new length with zeros (silence).

Value

For `duration`, the duration of the sample in seconds.

For `setDuration`, a Sample object with the new duration.

Author(s)

Matthias Heymann

See Also

`sampleLength`

Examples

```
## Not run:
s <- Sine(440,3)
duration(s)  # 3
duration(s) <-.5  # sample is now .5 sec long
play(setDuration(s,1)) # plays a .5 sec sine wave and then .5 sec silence
## End(Not run)
```

`fitSampleParameters` *Adjust Parameters of Two Sample Objects.*

Description

Adjust the parameters sampling rate, channels and bits/sample of two Sample objects.

Usage

```
fitSampleParameters(s1, s2)
```

Arguments

`s1,s2`	a Sample object, or strings giving the name of a wav file.

Details

The commands `rate`, `channels` and `bits` are used to transform copies of `s1` and `s2` to samples with the same parameters rate, channels and bits. Always the parameter with the better quality is chosen for the returned samples, that is the higher sampling rate, the larger number of channels and the larger number of bits per sample.

Value

a list containing the two transformed Samples as components.

Note

This routine is called before certain commands such as `sum.Sample` or `appendSample` are applied to Sample objects with different parameters.

Author(s)

Matthias Heymann

See Also

`rate`, `channels`, `bits`

Examples

```
## Not run:
s1 <- Sine(440,1,rate=22050,channels=1,bits=16)
s2 <- Sawtooth(440,1,rate=44100,channels=2,bits=8)
play(s1)
play(s2)
l  <- fitSampleParameters(s1,s2)
t1 <- l[[1]]
t2 <- l[[2]]
print(t1)
print(t2)  # both samples have the same parameters now
play(t1)
play(t2)   # none of the samples sounds different now,
           # since only parameters with higher quality were chosen
## End(Not run)
```

---

left *Extract one Channel from a Stereo Sample*

Description

Extract either the left or the right channel of a stereo Sample object or a stereo wav file.

Usage

```
left(s)
right(s)
```

Arguments

s	a Sample object, or a string giving the name of a wav file.

Details

If s is a mono sample, it will be returned as it is.

Value

a Sample object containing the left or the right channel of s.

Author(s)

Matthias Heymann

See Also

stereo for creating a stereo Sample object from two mono samples.

Examples

```
## Not run:
sLeft <- Sine(440,1)
sRight <- Sine(220,1)
s <- stereo(sLeft,sRight)
play(s)
play(left(s))  # only the left channel
play(right(s)) # only the right channel
## End(Not run)
```

---

loadSample *Load a WAV File from Disk*

Description

Load a wav file from disk and create a Sample object.

Usage

```
loadSample(filename, filecheck=TRUE)
```

Arguments

filename	a string giving the path and the name of the wav file.
filecheck	logical. If FALSE, no check for existance and read permission of the file will be performed.

Details

All kinds of wav files are supported: mono / stereo, 8 / 16 bits per sample, 1000 to 48000 samples/second.

Value

the Sample object that is equivalent to the wav file.

Note

`filename` can also be a Sample object. In this case, the same object will be returned immediately. This can be useful when writing functions that accept both Sample objects and the names of a wav file as an argument. See `is.Sample` for an example.

Author(s)

Matthias Heymann

See Also

`saveSample`, `as.Sample`

Examples

```
## Not run:
s <- loadSample("soundfile.wav")
play(s)
## End(Not run)
```

`mirror` *Mirror a Stereo Sample*

Description

Interchange the left and the right channel of a stereo Sample object or a stereo wav file.

Usage

```
mirror(s)
```

Arguments

`s`	a Sample object, or a string giving the name of a wav file.

Details

If `s` is a mono sample, it will be returned as it is.

Value

a Sample object, with the left and the right channel of `s` interchanged.

Author(s)

Matthias Heymann

See Also

`panorama` for a command with more parameters, `left` and `right` for access to single channels of a sample.

Examples

```
## Not run:
s <- stereo(Sine(440,1),Sine(220,1))
play(s)  # higher tone is on the left
play(mirror(s))  # now higher tone is on the right
## End(Not run)
```

noSilence *Cut Off Silence from a Sample Object*

Description

Cut off silence or low noise at the beginning and/or at the end of a Sample object or a wav file.

Usage

```
noSilence(s, level=0, start=TRUE, end=TRUE)
```

Arguments

s	a Sample object, or a string giving the name of a wav file.
level	non-negative numeric. Absolute values in the waveform matrix smaller than or equal to this value are regarded as silence.
start	logical. If TRUE, silence at the beginning of the sample will be cut off.
end	logical. If TRUE, silence at the end of the sample will be cut off.

Details

For stereo samples, it is checked if the values of both channels are silence before the silence is cut off.

Value

a Sample object without those parts at the start and at the end of the original sample that are below the specified noise level.

Author(s)

Matthias Heymann

See Also

cutSample

Examples

```
## Not run:
s <- Sine(440,5)
sound(s) <- sound(s)*matrix(seq(1,0,length=5*44100),nrow=1)
sampleLength(s)
play(s) # fade out
s <- noSilence(s,level=.05)
sampleLength(s)  # s is shorter now
play(s)    # although you don't hear that the end is missing
## End(Not run)
```

`normalize` *Rescale the Range of a Sample to [-1,1]*

Description

Multiply the waveform of a Sample object or a wav file with a positive constant so that the maximum absolut value becomes 1, or any other specified constant.

Use this command before saving or playing a Sample object to avoid cracks in the sound caused by parts in the waveform that exceed the range [-1,1].

Usage

```
normalize(s, level=1)
```

Arguments

`s`	a Sample object, or a string giving the name of a wav file.
`level`	a number between 0 and 1 specifying the desired maximum absolute value of the waveform.

Value

a Sample object with the specified maximum absolut value of the waveform.

Author(s)

Matthias Heymann

See Also

`Ops.Sample`, `center`

Examples

```
## Not run:
s <- .6*Sine(440,1)
plot(s)
plot(normalize(s)) # now it uses the full range
play(s)
play(normalize(s)) # this one is louder
## End(Not run)
```

`nullSample` *The NULL Sample Object*

Description

Create a Sample object whose waveform has length 1 and value 0. Often useful to initialize loops.

Usage

```
nullSample(rate=44100, bits=16, channels=1)
```

Arguments

`rate`	the sampling rate, between 1000 and 48000.
`bits`	the sample quality (number of bits per sample), 8 or 16.
`channels`	1 for mono, or 2 for stereo.

Value

a Sample object.

Note

Future versions may use a special NULLSample flag instead of using a sample of length 1.

Author(s)

Matthias Heymann

See Also

`Silence`

Examples

```
## Not run:
scale <- 2^(seq(0,1,length=13))[c(1,3,5,6,8,10,12,13)]
base <- 440
s <- nullSample()
for (f in scale)
  s <- appendSample(s,Sine(f*base,1))
play(s)
## End(Not run)
```

panorama *Narrow the Panorama of a Stereo Sample*

Description

Narrow the panorama of a stereo Sample object or of a stereo wav file.

Usage

```
panorama(s, pan)
```

Arguments

s	a Sample object, or a string giving the name of a wav file.
pan	a number between -50 and 50 giving the width of the panorama.

Details

If `abs(pan)<50`, mixtures of the two channels of `s` are used for the left and the right channel of the returned Sample object, so that they appear closer to the center. For `pan=0`, both sounds are completely in the center.

If `pan<0`, the left and the right channel are interchanged afterwards.

Value

a Sample object with the transformed panorama.

Author(s)

Matthias Heymann

See Also

`mirror` for `pan=-50`, `left` and `right` for access to single channels of the sample.

Examples

```
## Not run:
s <- stereo(Sine(440,1),Sine(330,1))
play(s)
play(panorama(s,30))  # now right and left tones are closer to the center
play(panorama(s,10))  # now even closer
play(panorama(s,0))   # now both at the center, the same as setChannels(s,1)
play(panorama(s,-30)) # again wider, but both sides switched
play(panorama(s,-50)) # the same as mirror(s)
## End(Not run)
```

`pitch` *Pitch a Sample Object*

Description

Change the pitch of a Sample object or a wav file.

Usage

```
pitch(s, semitones)
```

Arguments

`s`	a Sample object, or a string giving the name of a wav file.
`semitones`	a double giving the number of semitones to be pitched.

Details

Pitching +12 semitones (+1 octave) means to double the frequencies. Negative values of `semitones` are used to lower the frequencies.

Note that this transformation changes the actual data of the sample. Since pitching a sample is equivalent to playing it at a different speed, the length of the Sample object will also change.

Value

a Sample object.

Note

Future versions of this command may use a different algorithm to increase the quality of the returned sample.

Author(s)

Matthias Heymann

Examples

```
## Not run:
s <- Sine(440,1)
# Now play it 12 semitones = 1 octave deeper,
# that is half the frequencies and twice the length,
# or played at half speed.
play(pitch(s,-12)) # is the same as...
play(Sine(220,2))
## End(Not run)
```

play *Play a Sample Object or a WAV File*

Description

Play a Sample object or a wav file, using the wav file play command returned by `WavPlayer`, or any other specified command.

Usage

```
play(s, stay=FALSE, command=WavPlayer())
```

Arguments

`s`	a Sample object, or a string giving the name of a wav file.
`stay`	logical. If TRUE, the Windows Media Player won't be closed after playing the sample.
`command`	a character string giving the system command to be used for playing the sample.

Details

If `s` is a Sample object, it will be saved into a temporary folder before it is played. The temporary file will only be deleted afterwards if `stay=FALSE`.

Author(s)

Matthias Heymann

Examples

```
## Not run:
s <- Sine(440,1)
play(s)
## End(Not run)
```

plot.Sample *Plot a Sample Object*

Description

Plot the waveform of a Sample object or a wav file.

Usage

```
## S3 method for class 'Sample':
plot(x,xlab="sample #",ylab=NULL,...)
```

Arguments

x	a Sample object, or a string giving the name of a wav file. If x is a string, the explicit form plot.Sample must be used.
xlab	the character string giving the label for the x-axis.
ylab	For mono Sample objects as usual. For stereo Sample objects, ylab can be a vector of two strings to distinguish the y-labels for the left and the right channel. If ylab=NULL, the presets are used, that is "waveform" for mono samples and c("left","right") for stereo samples.
...	further graphical parameters.

Note

Use plot(s[interval]) to plot parts of s only (see examples).

If the range of the graph exceeds [-1,1], you can use the normalize command before plotting to get a better view of the waveform. (Then you should also call this function to avoid cracks in the sound before you save or play it the next time.)

Author(s)

Matthias Heymann

See Also

print.Sample

Examples

```
## Not run:
s <- Sine(440,1) + .4*Sine(1000,1)
plot(s[1:1000])
play(s)
s <- normalize(s)
plot(s[1:1000])  # now the range of the waveform is in [-1,1]
play(s)  # no cracks!
## End(Not run)
```

---

print.Sample *Print a Sample Object*

Description

Display the basic information about a Sample object or a wav file.

Usage

```
print.Sample(x,...)
```

Arguments

x	a Sample object, or a string giving the name of a wav file.
...	further parameters, not used at the moment.

Author(s)

Matthias Heymann

See Also

plot.Sample for plotting the waveform of a sample.

Examples

```
## Not run:
s <- Sine(440,1)
print(s)
## End(Not run)
```

---

rate *The Sampling Rate*

Description

Get or set the sampling rate (number of samples per second) of a Sample object or a wav file.

Usage

```
rate(s)
rate(s) <- value
setRate(s,value)
```

Arguments

s	a Sample object, or a string giving the name of a wav file.
value	an integer between 1000 and 192000 giving the sampling rate.

Details

The replacement form can be used to reset the sampling rate. Here, filenames are not accepted.

Note that changing the sampling rate of a Sample object affects the waveform of the sample.

Value

For rate, the sampling rate (number of samples per second) of the sample.

For setRate, a Sample object with the new sampling rate.

Note

Common sampling rates are between 8000 and 44100 (CD quality). Higher-quality recorders typically work with sampling rates of 48000, 92000 or 192000. Not every rate is guaranteed to be supported by every wav file player.

Future versions may use a different algorithm for sampling rate conversion to achieve a better sound quality for the returned sample.

Author(s)

Matthias Heymann

See Also

`fitSampleParameters`, `pitch`

Examples

```
## Not run:
s <- Sine(440,1,rate=44100)
rate(s)  # 44100
play(s)
print(s)
rate(s) <- 8000
play(s)  # s has worse quality now (noise and additional high frequencies)
print(s) # but uses less memory
## End(Not run)
```

---

reverse *Play a Sample Object Backwards*

Description

Returns the Sample object (or wav file) played backwards.

Usage

```
reverse(s)
```

Arguments

s	a Sample object, or a string giving the name of a wav file.

Value

a Sample object with the same parameters but with the sound played backwards.

Author(s)

Matthias Heymann

Examples

```
## Not run:
waveform <- 2*((seq(0,80,length=88200)%%1^2)-.5)
s <- as.Sample(waveform,44100,16)
play(s)
play(reverse(s)) # now played backwards
## End(Not run)
```

---

sampleLength *Length of a Sample Object*

Description

Get or set the length (number of columns in the waveform matrix) of a Sample object or a wav file.

Usage

```
sampleLength(s)
sampleLength(s) <- value
setSampleLength(s,value)
```

Arguments

s	a Sample object, or a string giving the name of a wav file.
value	an integer giving the sample length (number of columns in the waveform matrix).

Details

The replacement form can be used to reset the sample length (here, filenames are not accepted).

If a Sample object is shortened, extra values are discarded. When a Sample object is lengthened, it is padded out to its new length with zeros (silence).

Value

For `sampleLength`, the number of columns in the waveform matrix of the sample.

For `setSampleLength`, a Sample object with the new length.

Author(s)

Matthias Heymann

See Also

`duration`

Examples

```
## Not run:
s <- Sine(440,3,rate=44100,bits=16,channels=2)
sampleLength(s)  # 132300 samples ( = 3 sec * 44100 samples/sec )
sampleLength(s) <- 22050  # sample is now .5 sec long
play(setSampleLength(s,44100)) # plays a .5 sec sine wave and then .5 sec silence
## End(Not run)
```

---

saveSample *Save a Sample Object as a WAV File*

Description

Save a Sample object to disk as a wav file.

Usage

```
saveSample(s, filename, overwrite=FALSE)
```

Arguments

s	a Sample object.
filename	a string giving the path and the name of the destination file.
overwrite	logical. If FALSE and filename already exists, an error will be reported. Otherwise the old file will be deleted.

Author(s)

Matthias Heymann

See Also

loadSample

Examples

```
## Not run:
s <- Sine(440,1)
saveSample(s,"sine.wav")
## End(Not run)
```

`sound` *The Waveform Matrix of a Sample Object*

Description

Get or set the waveform matrix of a Sample object or a wav file.

Usage

```
sound(s)
sound(s) <- value
```

Arguments

`s`	a Sample object, or a string giving the name of a wav file.
`value`	a `channels(s)` x `sampleLength(s)` matrix of doubles.

Details

The replacement form can be used to reset the waveform of a sample object. Here, filenames are not accepted for codes.

The matrix can have one (for mono samples) or two rows (for stereo samples), where in the latter case the first row corresponds to the left and the second row to the right channel.

It contains the waveform(s) of the Sample object as sequence(s) of numbers between -1 and 1. `waveform` can contain arbitrary real numbers, but when the Sample object is played or saved to disk, [-1,1] is regarded as the native range of the sample, and any values outside this interval will cause cracks in the sound.

The waveform of a Sample object might exceed this interval during calculations. It is the task of the programmer to take care about the range of the waveform before saving or playing the sample, for example by using the `normalize` function.

Value

the waveform matrix of the sample.

Author(s)

Matthias Heymann

See Also

`as.Sample`

Examples

```
## Not run:
s <- Sine(440,1,channels=2)  # stereo sine wave
sound(s)[2,] <- sound(s)[2,]*seq(1,0,length=sampleLength(s))
play(s)  # right channel fades to zero
## End(Not run)
```

---

stereo *Create a Stereo Sample Object from Two Mono Samples*

Description

Create a stereo Sample object, given the two channels as Sample objects or wav files.

Usage

```
stereo(sLeft, sRight, pan=50)
```

Arguments

sLeft	a Sample object or a string giving the name of a wav file. Used for the left channel.
sRight	a Sample object or a string giving the name of a wav file. Used for the right channel.
pan	a number between -50 and 50 describing the distance between the two sound sources.

Details

If `abs(pan)<50`, mixtures of the two sources are used for the left and the right channel so that they appear closer to the center. For `pan=0`, both sounds are at the center. If `pan<0`, left and right channel are interchanged afterwards.

If the samples have different sample parameters (bits, rate and channels), the command codefitSampleParameters is called to adjust them before the two samples are combined.

Value

a stereo Sample object.

Author(s)

Matthias Heymann

See Also

`left`, `right`, `as.Sample`, `panorama`

Examples

```
## Not run:
sLeft <- Sine(440,1)
sRight <- Sine(220,1)
s <- stereo(sLeft,sRight)
play(s)
## End(Not run)
```

Index