Package 'audio'

November 25, 2021

Version 0.1-10

Title Audio Interface for R

Author Simon Urbanek <simon.urbanek@r-project.org>

Maintainer Simon Urbanek <simon.urbanek@r-project.org>

Depends R (>= 2.0.0)

Description Interfaces to audio devices (mainly sample-based) from R to allow recording and playback of audio. Built-in devices include Windows MM, Mac OS X AudioUnits and PortAudio (the last one is very experimental).

License MIT + file LICENSE

URL http://www.rforge.net/audio/

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-11-25 16:46:08 UTC

R topics documented:

audio.drivers *Audio Drivers*

Description

audio.drivers lists all currently loaded and available audio drivers.

current.audio.driver returns the name of the currently active audio driver or NULL if no audio drivers is available.

set.audio.driver selects an audio driver as the current driver.

load.audio.driver attempts to load a modular audio driver and, if succeessful, makes it the current audio driver.

Usage

```
audio.drivers()
current.audio.driver()
set.audio.driver(name)
load.audio.driver(path)
```

Arguments

name	name of the driver to load (as it appears in the name column of audio.drivers()) or NULL to load the default audio driver
path	path to the dynamic module to load

Details

The audio package comes with several built-in audio drivers (currently "wmm": WindowsMultiMedia for MS Windows, "macosx": AudioUnits for Mac OS X and "portaudio": PortAudio for unix), but it also supports 3rd-party drivers to be loaded (e.g. from other packages).

All operations that create new audio instances (play and record) use the current audio driver. The audio package allows the user to switch between different audio drivers. Each audio instance is tied to the driver it was created with even if the current driver was changed in the meantime.

Drivers are references by its short name listed in the name column in the list of available drivers (obtainable via audio.drivers).

An audio driver is any shared module that provides a C function create_audio_driver which returns a pointer to a populated structure audio_driver as defined in driver.h.

Value

audio.drivers returns a data frame lising all availbale drivers

set.audio.driver and current.audio.driver return the name of the active driver or NULL if no drivers ar avaliable.

load.audio.driver returns the name of the loaded driver.

See Also

record, play

Examples

```
audio.drivers()
```

---

audioInstance-methods *Audio instance class methods*

Description

audioInstances supports most methods relevant to them. See the corresponding generics help for details.

Noteworthy is that $data is the canonical way to get data associated with an audio instance - i.e. played or recorded content.

Usage

```
## S3 method for class 'audioInstance'
x$name
## S3 method for class 'audioInstance'
resume(x, ...)
## S3 method for class 'audioInstance'
pause(x, ...)
## S3 method for class 'audioInstance'
rewind(x, ...)
## S3 method for class 'audioInstance'
close(con, ...)
## S3 method for class 'audioInstance'
play(x, ...)
## S3 method for class 'audioInstance'
print(x, ...)
```

Arguments

x	audio instance
name	name of the attribute - currently only "data" is supported
con	audio instance (the name is unfortuantely defined in the close generic like this)
...	ignored

audioSample *Audio sample class and constructor*

Description

audioSample is a class encapsulating digitalized (sampled) audio data. Essentiually it tags numeric vectors and matrices with additional audio information (most importantly sample rate).

audioSample is the designated constructor of such objects.

Instances of the audioSample are numeric vectors or matrices with the following additional attributes:

- ratesample rate (in Hz), mandatory
- bitsresolution of the source (in bits), optional

If the object itself is a vector, it contains only one channel. Otherwise it is a matrix with as many rows as there are channels (hence 2 for stereo).

as.audioSample generic converts an object into an audio sample object. The default method is very similar to the constructor except that it attempts to infer the parameters from the object's attributes if they are not specified. Thus they are optional although they don't have visible defaults.

Usage

```
  audioSample(x, rate=44100, bits=16, clip = TRUE)
  as.audioSample(x, ...)
  ## Default S3 method:
as.audioSample(x, rate, bits, clip, ...)
  ## S3 method for class 'Sample'
as.audioSample(x, ...)
```

Arguments

x	object to convert or initilize with
rate	sample rate
bits	resolution of the source. It doesn't affect the data itself and is only used for playback and export.
clip	boolean value determining whether the source should be clipped to a range between -1 and 1. Values outside this range result in undefined behavior.
...	parameters passed to the object-specific method

Value

audioSample and as.audioSample return an audio sample object.

Examples

```
x <- audioSample(sin(1:8000/10), 8000)

play(x)
```

`audioSample-methods` *Audio sample class methods*

Description

`audioSample` methods behave in the same way as the underlying methods of numeric vectors and matrices except that they preserve the attributes and class of the objects.

Usage

```
  ## S3 method for class 'audioSample'
x$name
  ## S3 replacement method for class 'audioSample'
x$name <- value
  ## S3 method for class 'audioSample'
x[..., drop = FALSE]
  ## S3 method for class 'audioSample'
as.Sample(x, ...)
  ## S3 method for class 'audioSample'
print(x, ...)
```

Arguments

`x`	sample object
`name`	name of the attribute to get/set
`value`	value to set
`drop`	see `` `[` `` operator documentation
`...`	parameters passed to the object-specific method

Examples

```
x <- audioSample(sin(1:8000/10), 8000)
x$rate
x[1:10]
```

controls *Control audio instance*

Description

`pause` pauses (stops) audio recording or playback

`rewind` rewinds audio recording or playback, i.e., makes the beginning of the source (or target) object the current audio position.

`resume` resumes previously paused audio recording or playback

Usage

```
pause(x, ...)
rewind(x, ...)
resume(x, ...)
```

Arguments

`x`	instance object
`...`	optional arguments passed to the method specific to the object

Value

All functions return TRUE on success and FALSE on failure. All methods are generics and intended to apply to similar asynchronous operations (e.g. movie playback etc.).

See Also

`play`, `record`

play *Play audio*

Description

`play` plays audio

Usage

```
play(x, ...)
## S3 method for class 'audioSample'
play(x, rate, ...)
## S3 method for class 'Sample'
play(x, ...)
## Default S3 method:
play(x, rate = 44100, ...)
```

Arguments

x	data to play
rate	sample rate - it is inferred from the object (where possible) if not specified
...	optional arguments passed to the method specific to the object being played

Value

Returns an audio instance object which can be used to control the playback subsequently.

Examples

```
play(sin(1:10000/20))
```

---

record *Record audio*

Description

record record audio using the current audio device

Usage

```
record(where, rate, channels)
```

Arguments

where	object to record into or the number of samples to record
rate	sample rate. If ommitted it will be taken from the where object or default to 44100
channels	number of channels to record. If ommitted it will be taken from the where object or default to 2. Note that most devices only support 1 (mono) or 2 (stereo).

Details

The record function creates an audio instance of the current audio driver to start audio recording. The recording is performed asynchronously and the function returns immediately after the recording is started.

where can either be a numeric vector of the storage mode 'double' and length greater than 1 or a numberic vector of length one specifying the number of samples to record. The the former case the audio data is recorded directly to the vector, in the latter case a new object (initialized with NA) is created internally (and thus only accessible using a$data where a is the audio instance).

The recording is automatically stopped after the where object has been completely filled. Nonetheless pause can be used to stop the recoding at any time.

Value

Returns an audio instance object which can be used to control the recording subsequently.

Examples

```
x <- rep(NA_real_, 16000)
# start recording into x
record(x, 8000, 1)
# monitor the recording progress
par(ask=FALSE) # for continuous plotting
while (is.na(x[length(x)])) plot(x, type='l', ylim=c(-1, 1))
# play the recorded audio
play(x)
```

---

wait *Wait for an event*

Description

`wait` waits until an event occurs or times out

Usage

```
wait(x, ...)
## Default S3 method:
wait(x, timeout, ...)
## S3 method for class 'audioInstance'
wait(x, timeout=NA, ...)
```

Arguments

`x`	event to wait for
`timeout`	longest period to wait for (in seconds, real number). A value of 0 causes `wait` to just check for the event, values `NA` and less than zero mean to wait indefinitely until the even occurs.
`...`	optional arguments passed to the method specific to the object being monitored

Details

The default method allows `x` to specify the timeout, i.e., if `timeout` is not specified and `x` is numeric then the timeout is set to `x`.

Value

Returns the result.

Examples

```
# play a sound and wait until the playback is done
wait(play(sin(1:10000/20)))
# wait for 2.5 seconds unconditionally
wait(2.5)
```

---

wave *WAVE file manipulations*

Description

`load.wave` loads a sample from a WAVE file

`save.wave` saves a sample into a WAVE file

Usage

```
load.wave(where)
save.wave(what, where)
```

Arguments

`where`	file name of the file to load from or save to
`what`	audioSample object to save

Details

WAVE is a RIFF (Resource Interchange File Format) widely used for storage of uncompressed audio data. It is often identified by the extension .WAV on DOS-legacy systems (such as Windows). Although WAVE files may contain compressed data, the above functions only support plain, uncompressed PCM data.

Value

`load.wave` returns an object of the class `audioSample` as loaded from the WAVE file

`save.wave` always returns `NULL`

See Also

`audioSample`, `play`, `record`

Index