Package 'csound'

July 2, 2014

Type Package

Title Accessing Csound functionality through R

Version 0.1-1

Date 2012-04-30

Author Ethan Brown

Maintainer Ethan Brown <ethancbrown@gmail.com>

Depends R (>= 2.12.0), rdyncall

Description Provides basic access in an R session to Csound <http://www.csounds.com>, a powerful free and open-source software sound synthesizer. The package functionality is largely geared towards supporting the needs of the playitbyr package for sonification and is not particularly mature on its own. But it certainly can be used without knowing anything about playitbyr.

SystemRequirements Csound (>= 5), available at <http://csound.sourceforge.net/>. This package uses the shared library file, which is included with standard Mac and Windows installations of Csound. Some Linux distributions package the library file separately from the executible, so for instance on Debian and Ubuntu you would need the ``libcsound64-dev'' package. See the package web site for more guidance on setup and configuration.

URL http://playitbyr.org/csound.html

License GPL (>= 2)

LazyLoad Yes

Collate 'aaa.R' 'setCsoundLibrary.R' 'getCsoundError.R' 'getHeaderInfo.R' 'PerformanceAPI.R' 'createPerformance.R' 'writeCsoundScore.R' 'scoreMatrices.R'

Repository CRAN

Date/Publication 2012-04-30 19:11:28

NeedsCompilation no

R topics documented:

`createPerformance` *Perform score statements with specified Csound orchestra*

Description

These functions provide a safe, streamlined, and R-friendly interface to Csound's API, allowing users to pass lists of matrices and vectors to use as parameter data for controlling Csound instruments.

Usage

```
createPerformance(i = NULL, f = NULL,
  orcfile = "built-in.orc", scorefile = NULL,
  out = "dac", realTime = FALSE,
  finishPerformance = TRUE, suppressDisplays = TRUE,
  moreflags = NULL)

performScoreRealTime(csInstance, i = NULL, f = NULL)

finishPerformance(csInstance)

cleanupCrash()
```

Arguments

`i`	A list of `matrix` objects. Each `matrix` is the instructions for a single instrument. Each row of the `matrix` is an `i` statement, which instructs Csound to make an instrument active at a specific time and for a certain duration, and with certain parameters (p-fields). These p-fields are interpreted in the order of the columns of the `matrix`.
`f`	A list of numeric vectors; these create the function tables Csound uses for oscillators and various other uses.
`orcfile`	The path of the orchestra file to be used for the performance. If this equals `"built-in.orc"`, the default, the orchestra included with this package will be used (see `scoreMatrices` for more details of using the built-in instruments.)
`scorefile`	The path of the score file, if any, to be used for the performance. The whole purpose of this function is to feed the score statements to Csound and bypass the need for score files, but this option is provided in any case.

`out` String representing where to send output sound; the default, `"dac"`, indicates to send it your computer's sound output. If you want to render a file, enter the path to the (WAV) file you want.

`realTime` Indicates whether the performance is to be rendered in real time. If you are rendering to a file, you probably want this as `FALSE`, since it can render a whole lot faster than real-time to file.

`finishPerformance`
Should the performance be closed after completing the score? If `TRUE`, the default, cleans up and closes Csound. If `FALSE`, returns a pointer to a Csound instance that can be used to continue the performance or eventually close it.

`suppressDisplays`
Csound by default pops up with annoying graphical widgets. This alloys you to suppress them (the default).

`moreflags` A character vector of extra command-line flags to pass to Csound upon compilation of the orchestra. See The Csound Manual's page on the Csound command-line options.

`csInstance` An instance of Csound that can be used to continue or close the current performance.

Details

`createPerformance()` initializes Csound and compiles the orchestra and allows the rendering of matrices and vectors as parameter data; it can either close its instance of Csound or return it, depending on the value of the argument `finishPerformance`.

If `finishPerformance = FALSE`, one can use the object returned by `createPerformance()` to control Csound and subsequently add more control data which Csound then performs immediately with `performScoreRealTime()`. You can then finish up the performance and close Csound with `finishPerformance()`.

If you encounter an error and cannot run `finishPerformance()` run `cleanupCrash` before doing anything else. Otherwise you may cause a segfault and crash R.

See Also

`scoreMatrices()` for easy creation of the `i` argument to `createPerformance`

Examples

```
sndcheck <- scoreMatrices(5, 5)
sndcheck$FM[, "start"] <- 0:4
sndcheck$FM[, "dur"] <- 0.5
sndcheck$FM[, "amp"] <- 0.5
sndcheck$FM[, "pan"] <- (0:4)/4
sndcheck$FM[, c("attkp", "decayp")] <- 0.01
sndcheck$FM[, "cps"] <- (1:5)*110
sndcheck$FM[, "mod"] <- (1:5)/2
sndcheck$FM[, "indx"] <- 4:0
sndcheck$subtractive[, "start"] <- 0:4 + 0.5
sndcheck$subtractive[, "dur"] <- 0.1
```

```
sndcheck$subtractive[, "amp"] <- 0.05
sndcheck$subtractive[, "pan"] <- (4:0)/4
sndcheck$subtractive[, c("attkp", "decayp")] <- 0.01
sndcheck$subtractive[, "cntr"] <- (5:1)*500
sndcheck$subtractive[, "bw"] <- (5:1)*500
## Not run: createPerformance(sndcheck)
```

`getCsoundLibrary` *Get and set the Csound shared library for accessing Csound's functionality.*

Description

Functions to get and set the option `"csoundlibrary"`, which is used by all the functions in the package to actually access the functionality of Csound.

Usage

```
getCsoundLibrary()

setCsoundLibrary(path)

.csoundGetVersion()
```

Arguments

`path`	A character string containing the path to the Csound shared library.

Details

The **csound** package attempts to find the library file automatically on startup and warns if it fails. In this case, you need to find the path to Csound's shared library (often a `.so`, `.dll` or `.dylib`, depending on the system). `setCsoundLibrary` will only actually set the option if it successfully locates and links to the library; if it fails to link it, it stops with an error.

Hopefully these options won't need to be accessed by most users, but they are provided as a backup.

Value

`getCsoundLibrary()` returns a pointer to the Csound library, with the path to the library passed as an attribute.

`.csoundGetVersion()` returns the version of Csound that is linked to; this is useful as a quick check to make sure everything has linked correctly.

See Also

`getHeaderInfo`

getHeaderInfo *Get the header specifications of a Csound instance*

Description

The header of the csound orchestra declares various important variables including the sample rate, the number of audio samples per control period, the number of output channels, and the maximum amplitude value. This function queries a Csound instance and returns the information in a list.

Usage

```
getHeaderInfo(csInstance)
```

Arguments

csInstance	An instance of Csound, created by .csoundCreate.

Value

getHeaderInfo() returns a list with the following elements:

$sample rate, abbreviated as sr in the orchestra header. is the number of data points ('samples') per second used to represent a sound. The default is 44100 Hz, which is CD-quality.

$ksmps is the number of samples in a control period, which must be an integer. Csound allows one to control an audio signal at a rate below that of the audio rate (i.e. the *control rate* or $kr), useful for things such as creating an envelope for sounds; using a lower-than-audio rate uses less processing power for what is often the same audio effect. Higher values of ksmps indicate a slower control rate; ksmps = 1 means audio rate. A typical ksmps is 10.

$x0dbfs is the maximum amplitude value–specified amplitude values are scaled between 0 and this number. 0dbfs is short for Zero Decibels at Full-Scale Amplitude. A common value is 1; if not specified in the file, the default is 32768.

$nchnls is the number of output channels specified; 1 is mono output, 2 stereo, etc.

If the orchestra has been compiled, these should match the header in the orchestra; if no orchestra has been compiled, getHeaderInfo() simply returns the defaults.

scoreMatrices *Create blank score matrices for the built-in instruments*

Description

This convenience function creates score matrices (for use in createPerformance with the instruments included with the **csound** package.

Usage

```
scoreMatrices(nFM = 0, nsubtractive = 0)
```

Arguments

`nFM`	The number of rows (each representing a note) desired for the FM synthesis instrument.
`nsubtractive`	The number of rows (each representing a note) desired for the subtractive synthesis instrument.

Details

There are currently two instruments included with the **csound** package: a basic FM synthesis instrument and a subtractive synthesis instrument.

Value

A list of two matrices: `$FM`, the score events for the FM instrument, and `$subtractive`, the score events for the subtractive synthesis instrument. These can be used as the `i` argument to `createPerformance()`.

Parameters for built-in instruments

The columns of the returned matrices are parameters that Csound uses to render each note. The two instruments share several parameters:

instr The instrument number. `scoreMatrices()` fills this in for you and users should not need to alter this.

start The starting time of the note (in seconds).

dur The duration of the note (in seconds).

amp The volume of the note, as a proportion between 0 and 1, where 1 is the maximum volume. Note that a multiple notes that happen at the same time could add up to more than one, causing distortion an clipping.

pan The stereo placement of the note; 0 means entirely on the left speaker, and 1 means entirely on the right.

attkp The proportion of the note's length devoted to the initial (linear) attack.

decayp The proportion of the note's length devoted to the (linear) decay.

The FM instrument has these additional parameters:

cps The frequency of the carrier tone (in Hertz).

mod The modulating frequency, given as a *multiple* of the carrier tone.

indx The index of modulation.

The subtractive synthesis instrument has these additional parameters:

cntr The central frequency of the band-pass filter (in Hertz).

bw The bandwidth of the band-pass filter (in Hertz).

Note

When using the built-in instruments with `createPerformance()`, these matrices can be used as the `i` argument and no `f` argument is needed.

Examples

```
sndcheck <- scoreMatrices(5, 5)
sndcheck
sndcheck$FM[, "start"] <- 0:4
sndcheck$FM[, "dur"] <- 0.5
sndcheck$FM[, "amp"] <- 0.5
sndcheck$FM[, "pan"] <- (0:4)/4
sndcheck$FM[, c("attkp", "decayp")] <- 0.01
sndcheck$FM[, "cps"] <- (1:5)*110
sndcheck$FM[, "mod"] <- (1:5)/2
sndcheck$FM[, "indx"] <- 4:0
sndcheck$subtractive[, "start"] <- 0:4 + 0.5
sndcheck$subtractive[, "dur"] <- 0.1
sndcheck$subtractive[, "amp"] <- 0.05
sndcheck$subtractive[, "pan"] <- (4:0)/4
sndcheck$subtractive[, c("attkp", "decayp")] <- 0.01
sndcheck$subtractive[, "cntr"] <- (5:1)*500
sndcheck$subtractive[, "bw"] <- (5:1)*500
sndcheck
## Not run: createPerformance(sndcheck)
```

`writeCsoundScore` *Write a Csound score file, given lists of i and f statements*

Description

Sometimes you want to write data to a Csound score file, either for rendering later or for when you want to render to a sound file. This is principally intended to be called by `createPerformance`.

Usage

```
writeCsoundScore(i, f, outfile = NULL)
```

Arguments

`i`	A list of `matrix` objects. Each `matrix` is the instructions for a single instrument. Each row of the `matrix` is an `i` statement, which instructs Csound to make an instrument active at a specific time and for a certain duration, and with certain parameters (p-fields). These p-fields are interpreted in the order of the columns of the `matrix`.
`f`	A list of numeric vectors; these create the function tables Csound uses for oscillators and various other uses.
`outfile`	The name of the file to write to. If codeNULL, the default, the score is written to a temporary file.

Value

The file name that the score was written to.

Index