Package: AIFFtools (via r-universe)

May 25, 2026

Type Package

Title Read AIFF Files and Convert to WAVE Format

Version 1.0

Date 2024-08-20-

Description Functions are provided to read and convert AIFF audio files to WAVE (WAV) format. This supports, for example, use of the 'tuneR' package, which does not currently handle AIFF files. The AIFF file format is defined in <https://web.archive.org/web/20080125221040/http://www.borg.com/~jglatt/tech/aiff.htm> and <https://www.mmsp.ece.mcgill.ca/Documents/AudioFormats/AIFF/Docs/AIFF-1.3.pdf>

License LGPL-3

Imports R.utils, methods, tuneR

NeedsCompilation no

Author Carl Witthoft [aut, cre]

Maintainer Carl Witthoft <cellocgw@gmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2024-09-06 02:52:59 UTC

RemoteUrl https://github.com/cran/AIFFtools

RemoteRef HEAD

RemoteSha 8db1ca18ecdf97d900bda05ecba248b5bd7db091

Contents

AIFFtools-package *Read AIFF Files and Convert to WAVE Format*

Description

Functions are provided to read and convert AIFF audio files to WAVE (WAV) format. This supports, for example, use of the 'tuneR' package, which does not currently handle AIFF files. The AIFF file format is defined in <https://web.archive.org/web/20080125221040/http://www.borg.com/~jglatt/tech/aiff.htm> and <https://www.mmsp.ece.mcgill.ca/Documents/AudioFormats/AIFF/Docs/AIFF-1.3.pdf> .

Details

The DESCRIPTION file:

Package:	AIFFtools
Type:	Package
Title:	Read AIFF Files and Convert to WAVE Format
Version:	1.0
Date:	2024-08-20-
Authors@R:	c(person(given = "Carl", family = "Witthoft", role = c("aut","cre"), email= "cellocgw@gmail.com"))
Description:	Functions are provided to read and convert AIFF audio files to WAVE (WAV) format. This supports, for examp
License:	LGPL-3
LazyData:	FALSE
Imports:	R.utils, methods, tuneR
Author:	Carl Witthoft [aut, cre]
Maintainer:	Carl Witthoft <cellocgw@gmail.com>

Author(s)

Carl Witthoft [aut, cre]

Maintainer: Carl Witthoft <cellocgw@gmail.com>

aiff2wav *Function to Convert an AIFF - Class Object to a Wave - or WaveMC - Class Object*

Description

This function converts an object produced with `readAiff` to a Wave-class object such as is generated with `readWave` .

Usage

```
aiff2wav(aifsrc,  makeMC = FALSE, ...)
```

Arguments

aifsrc	The source object. Must be of class "Aiff" as created with readAiff
makeMC	If TRUE, force the output to be of class WaveMC regardless of the number of input channels. However, if the input audio array has more than two channels, the value is always set to TRUE.
...	Reserved for future use

Details

Given that AIFF files are PCM encoded, the default value of the slot @pcm is TRUE and remains untouched. The slots @bits, @samp.rate are read from the source object. Note, however, that should the source bit-depth be a value unsupported by wave files, the function will exit with an error message. For class "Wave" , the slot @stereo is set to FALSE if the input is a vector or single column, and to TRUE if there are two columns.

Value

A Wave or WaveMC object as defined in the tuneR package.

Author(s)

Author and Maintainer:Carl Witthoft <carl@witthoft.com>

See Also

readAIFF to load an AIFF file.

Examples

```
samp3 <- readAiff(system.file("extdata", "sample3.aif", package = "AIFFtools") )
samp3wav <- aiff2wav(samp3)
```

---

readAiff *Function Which Reads An AIFF Audio File.*

Description

This function reads and parses an AIFF audio file, producing the audio data along with attached information such as the recording name, date, midi data, markers, etc. (most of these are optional and may not exist in the source file)

Usage

```
readAiff(filename, from = 1, to = Inf, ... )
```

Arguments

filename	Full path to the desired aif file, as a character string.
from	The first frame of audio to be returned Default is 1. Be aware that selecting a subset of the audio may cause other data such as 'Markers' or "Midi" to fail.
to	The last frame of audio to be returned Default is "Inf", meaning all frames. If to is larger than the data source, pads or empty frames are not added. Be aware that selecting a subset of the audio may cause other data such as 'Markers' or "Midi" to fail.
...	Reserved for future use

Details

The AIFF format specification, https://web.archive.org/web/20080125221040/http://www.borg.com/~jglatt/tech/aiff.htm, https://www.mmsp.ece.mcgill.ca/Documents/AudioFormats/AIFF/Docs/AIFF-1.3.pdf , allows multiple instances of some chunks such as MIDI and APPL. As of this revision, readAiff will only return the last of multiple chunks in a file. There are several versions of "ID3" chunk formats identified at https://id3.org/ . It is not known whether this function will handle all variations correctly.

Value

A list, consisting of at least the following. "COMM", "SSND", "MARK", "INST", "COMT", "AUTH", "(c), ", "ANNO", "AESD", "MIDI", "APPL", "ID3, ", "NAME", "haveData", "unknown" . If other chunk types are found, they will be added as a new list item.

COMM: a list with: $numChan the number of audio channels $numFrame the number of frames (one frame contains a single sample from every channel) $sampleSize number of bits per sample $sampleRate sample rate in samples per second

SSND: a list with: $offset the offset of the start of data frames as defined in the AIFF spec. $blocksize block size of data frames as defined in the AIFF spec. $audio an array, one channel per column, of the audio samples.

MARK: a list with: $MarkerId the ID numbers of the markers in use $Mposition the positions within the data frames of the markers $Mname the names of the markers

INST: a list of MIDI instrument information: $baseNote, $detune, $lowNote, $highNote, $lowVelocity, $highVelocity , $gain are numeric values . See the AIFF specification for details. $sustainLoop and $releaseLoop each contain three integers representing the "play mode," "begin loop markerID," and "end loop markerID."

COMT: a list with $nbr the number of comments $comments a vector of character strings AUTH: a character string of the author "(c) ": copyright notice ANNO: a comments string. AESD: quoting from the AIFF spec, "The 24 bytes of AESChannelStatusData are specified in the AES Recommended Practice for Digital Audio Engineering - Serial Transmission Format for Linearly Represented Digital Audio Data, section 7.1, Channel Status Data. That document describes a format for real-time digital transmission of digital audio between audio devices. This information is duplicated in the Audio Recording Chunk for convenience." MIDI: quoting from the AIFF spec, " The MIDI Data Chunk can be used to store MIDI data (please refer to Musical Instrument Digital Interface Specification 1.0, available from the International MIDI Association, for more details on MIDI)." APPL: a free-form block that can be used to store application-specific parameters.

"ID3 " a list with $IDver the ID3 version, main and sub, e.g. "3 0" $IDflag flag information as described in documents at https://id3.org $data an array with columns named "tag," "encode," "message," where the tag identifies the type of information and the encode value indicates whether the message text is ASCII or some variant of UTF-X. NAME: the name of the piece of music (in general). haveData: a character vector of all chunks found. All other chunks' list elements will contain only NULL. unknown: a character vector of chunk names that are not currently known to this function. Their data contents will be provided in list elements under these names. Other list elements, if any, will be named for the chunk names they represent.

Author(s)

Author and Maintainer:Carl Witthoft <carl@witthoft.com>

See Also

Aiff2wav to convert an uploaded AIFF data into a Wave or WaveMC class object as defined in the package tuneR

Examples

```
samp3 <- readAiff(system.file("extdata", "sample3.aif", package = "AIFFtools") )
```

Index