1. midi2wave()-function

The main Function is midi2wave(event.frame, ctrl) with two arguments: event.frame and ctrl.

- event.frame is a data frame which represents a midi schema for the music piece. It should be a data frame with following columns:
    - track.number: Number of the track (an integer number)
    - channel.number: Number of the channel (an integer number)
    - note.numer: Number of the note in MIDI-Notation (an integer number in [0,127]) (http://www.tonalsoft.com/pub/news/pitch-bend.aspx)
    - velocity: Velocity (an integer number in [1, 100])
    - start.time: note starting time
    - end.time: note ending time
    - message.number.note.on: an integer number for note.on message (unimportant for this application, could be NA's)
    - message.numer.note.off: an integer number note.off message (unimportant for this application, could be NA's or message.number.note.on+1)
- ctrl is an output of RealConverterControl-function. List of following elements:
    - channels: a vector of integer numbers. Should match with channels of the event.frame. Default is 0.
    - instruments: a string vector which provides a music instrument to each channel. Possible instruments are: c("piano", "guitar", "flute", "clarinet", "trumpet", "violin"). Default is "piano".
    - wave.to.channel: a logical parameter. If TRUE, a Wave object is generated for each channel, additionally to the composite Wave object. Default is FALSE
    - sample.instrument: a string vector which determines for each channel whether the a special fixed instrument should be taken, instrument should be sampled randomly from the available data set (mostly 2-3 note data sets for an instrument), so that just tones from the samples data based are used in the channel or each tone should be sampled randomly from all available data sets for the interesting instrument. The associated allowed values are: c("fix", "sample.instrument", "sample.each.tone").
    - fix.string: a logical parameter which is important just for the string instrument where some tones can be produced by different strings (like guitar). If TRUE, always the highest possibly string is chosen, if FALSE, the string is sampled randomly. Default is FALSE.
    - thresh.staccato: a time threshold in seconds for the decision whether a tone should be taken from the staccato data bank (short tons): if a duration of a tone is smaller than this threshold. Default: 0.15 seconds. Maximal value is 0.8 seconds.
    - data.path: a path for the directory with tone data banks. Important, if instrument paths are not specified. Default is "//STORE/share/LSWeihs/LSWeihsServer/DissBauer2015/RealConverterTones"
    - notes.path.piano: a path for piano tones data bank. If notes.path.piano = NULL (Defualt) and fix.string = “fix”, “.../piano/tr011PF” tone library is taken. If notes.path.piano = NULL and fix.string != “fix”, notes.path.piano = c(".../piano/tr011PF", ".../piano/tr012PF", ".../piano/tr013PF")
    - notes.path.guitar: a path for guitar tones data bank. If notes.path.guitar = NULL (Defualt) and fix.string = “fix”, “.../guitar/tr092CGRF” tone library is taken. If notes.path.guitar =

NULL and fix.string != “fix”, notes.path.guitar = c("…/guitar/tr092CGRF", "…/guitar/tr092CGRN", "…/guitar/tr092CGAF", "…/guitar/tr092CGAN", "…/guitar/tr093CGRF", "…/guitar/tr093CGRN", "…/guitar/tr093CGAF","…/guitar/tr093CGAN","…/guitar/tr091CGRF", "…/guitar/tr091CGRN", "…/guitar/ttr091CGAF", "…/guitar/tr091CGAN)

- notes.path.flute: a path for flute tones data bank. If notes.path.flute = NULL (Defualt) and fix.string = “fix”, “…/flute/tr331FL” tone library is taken. If notes.path.flute = NULL and fix.string != “fix”, notes.path.flute = c("…/flute/tr331FL", "…/ flute/tr332FL").
- notes.path.clarinet: a path for clarinet tones data bank. If notes.path.clarinet = NULL (Defualt) and fix.string = “fix”, “…/clarinet/tr311CL” tone library is taken. If notes.path.clarinet = NULL and fix.string != “fix”, notes.path.clarinet = c("…/flute/tr331FL", "…/ flute/tr332FL").
- notes.path.trumpet: a path for trumpet tones data bank. If notes.path.trumpet = NULL (Defualt) and fix.string = “fix”, “…/trumpet/tr211TR” tone library is taken. If notes.path.trumpet = NULL and fix.string != “fix”, notes.path.trumpet = c("…/trumpet/tr211TR", "…/ trumpet/tr212TR").
- notes.path.violin: a path for violin tones data bank. If notes.path.violin = NULL (Defualt) and fix.string = “fix”, “…/violin/tr151VN” tone library is taken. If notes.path.violin = NULL and fix.string != “fix”, notes.path.violin = c("…/violin/tr151VN", "…/violin/tr152VN", "…/violin/tr153VN",).

The output of midi2wave() function is a list with two elements:

- wave.object: an object of Wave-Class (see tuneR R-package), if ctrl$wave.to.channel = FALSE. Else, a list which number of channels + 1 elements. The last element is Wave-object of the composite wave.
- notes.path: a list. Each element corresponds to a channel and provides path’s to the notes which were used for wav generation in this channel.

```
### example 1

event.frame=readMidiMatlab("Flute1", "//STORE/share/LSWeihs/LSWeihsServer/DissBauer2015/Datenbank/Klassik")

ctrl=RealConverterControl(
        channels=unique(event.frame$channel.number),
         instruments = "flute",
        sample.instrument = "sample.instrument",
        data.path="//STORE/share/LSWeihs/LSWeihsServer/DissBauer2015/RealConverterTones")

wav.object=midi2wave(event.frame, ctrl)
```

2. Generating event.frame data frame

event.frame can be generated either by hand or (in most cases) using readMidiMatlab() function. For this reason the Matlab program should be installed as there does not exist any R package for reading MIDI until yet.

example 2a: In the first case an example can be given as follows:

```
event.frame = data.frame(track.number = c(1,1,1,1), channel.number= c(0, 0, 0, 0),
        note.number = c(74, 81, 77, 76), velocity = c(75, 80, 75, 73),
        start.time = c(0, 0.91, 1.09, 1.28), end.time = c(0.91, 1.09, 1.28, 1.46),
        message.number.note.on = c(15, 17, 19, 21), message.number.note.off = c(16, 18, 20, 22)
```

If using readMidiMatlab(file.name, file.direktory) function, a main data directory have first be generated with following subdirectories: mat, mid, midiOriginal, times and wav.

- file.name is the name of interesting music piece, without ending (just name)
- file.direktory is the path to the main directory

The subdirectory "file.directory /midiOriginal" should contain the original midi file: file.name.mid.

file.name.midi file will be produced by Matlab and saved in " file.directory /mid"

file.name.mat file will be produced by Matlab and saved in " file.directory /mat"

file.name.txt file containing true onset times will be saved in " file.directory /times"

example 2b:

```
event.frame = readMidiMatlab("Flute1", "//STORE/share/LSWeihs/LSWeihsServer/DissBauer2015/Datenbank/Klassik")
```

3. Generating ctrl object

Ctrl object should be generated using RealConverterControl() function. All arguments of this function are described in Sec. 1. Here some examples are provided:

example 3a: generate a flute wav while sampling the flute instrument randomly from the both available libraries: "…/flute/tr331FL" and "…/ flute/tr332FL".

```
ctrl=RealConverterControl(
        channels=0,
        instruments="flute",
        sample.instrument="sample.instrument",
        data.path="//STORE/share/LSWeihs/LSWeihsServer/DissBauer2015/RealConverterTones")
```

example 3b: use two channels: 0 for guitar and 1 for flute. For flute use the “tr331FL” library, while the library for guitar will be sampled randomly. Do not fix the string(important just for guitar). Return a separate Wave object for each channel.

```
ctrl=RealConverterControl(
        channels = c(0,1),
        instruments = c("guitar","flute"),
        wave.to.channel = TRUE,
        fix.string = FALSE,
        notes.path.flute = "//STORE/share/LSWeihs/LSWeihsServer/DissBauer2015/RealConverterTones/flute/tr331FL",
        sample.instrument="sample.instrument",
        data.path="//STORE/share/LSWeihs/LSWeihsServer/DissBauer2015/RealConverterTones")
```

4. Examples

```
### example 4a: monophonic flute

#load required libraries
library(tuneR)
library(R.matlab)

# source required R functions
source("HelpFunctions.R")
source("makeTrack.R")
source("extractNote.R")
source("midi2wave.R")
source("readwave2.R")
source("MakeRealConverterControl.R")
source("readMidiMatlab.R")

event.frame=readMidiMatlab("Flute1", "//STORE/share/LSWeihs/LSWeihsServer/DissBauer2015/Datenbank/Klassik")

ctrl=RealConverterControl(channels=unique(event.frame$channel.number),
                          instruments="flute",
                          sample.instrument="sample.instrument",
                          data.path="//STORE/share/LSWeihs/LSWeihsServer/DissBauer2015/RealConverterTones")

w=midi2wave(event.frame,ctrl)
# "do track for the instrument:" "flute"
w.o=w$wave.object
writeWave(wo,file="//STORE/share/LSWeihs/LSWeihsServer/DissBauer2015/Datenbank/Klassik/wav/trueFlute1.wav")
play(w.o)


### example 4b: flute and guitar
event.frame=readMidiMatlab("GuitarFlute1", "//STORE/share/LSWeihs/LSWeihsServer/DissBauer2015/Datenbank/Klassik")

# ckeck the channels
unique(event.frame$channel.number)
table(event.frame$channel.number)
#0   1
#112 108
# 0 is guitar
# 1 is flute

ctrl=RealConverterControl(channels=unique(event.frame$channel.number),
                          instruments=c("guitar","flute"),
                          wave.to.channel = TRUE,
                          notes.path.flute = "//STORE/share/LSWeihs/LSWeihsServer/DissBauer2015/RealConverterTones/flute/tr331FL",
                          sample.instrument="sample.instrument",
                          data.path="//STORE/share/LSWeihs/LSWeihsServer/DissBauer2015/RealConverterTones")

w=midi2wave(event.frame,ctrl)
wo=w$wave.object
writeWave(wo,file="//STORE/share/LSWeihs/LSWeihsServer/DissBauer2015/Datenbank/Klassik/wav/trueGuitarFlute1.wav")

play(wo[[1]]$channel.wave) # flute
play(wo[[2]]$channel.wave) # guitare
play(wo[[3]]) # composite wav
```