DSC 2001 Proceedings of the 2nd International Workshop on Distributed Statistical Computing
March 15-17, Vienna, Austria
http://www.ci.tuwien.ac.at/Conferences/DSC-2001
K. Hornik & F. Leisch (eds.) *ISSN 1609-395X*

R sings - or using R to sonify data

Erich Neuwirth[*]

Abstract

We present a new package which allows R to produce sound from data. The packages uses Csound, an extremely powerful and freely available multi platform program for sound creation. Sonification is a relatively new branch of data analysis, so even the basic set of concepts is still in flux. Our toolkit hopefully makes research into different options for sonification methods more accessible and helps to develop "data based instruments" in R.

1 Introduction

Advances in hardware have made it possible to synthesize complex sounds in real-time on easily available desktop computers. One of the most powerful software synthesizer programs is Csound, available from `ftp://ftp.maths.bath.ac.uk/dream/`.

On `http://mailbox.univie.ac.at/erich.neuwirth/Rcsound/` we supply an R package which makes it possible to program sounds from R, thereby extending the possible representations of data for R.

The package is still very much in development, one of the reasons being that sonification is an area of data analysis which does not have a canonical set of methods, and our package has been created with the intent of having a test bed for different approaches to these new possibilities.

2 Examples of use

After loading library `Rcsound`

[*]Department of Statistics and Decision Support Systems, University of Vienna

```
csndPlayMono(1,c(0:12),1,8000,264*2^(c(0:12)/12))
```

plays a sequence of 12 semitones from frequency 264 to frequency 528. The tone is played as a pure sine wave.

The header of the function is

```
csndPlayMono <- function(instr, start, dur, vol, freq)
```

and the meaning of the parameters is:

`instr`	instrument number
`start`	vector of starting times of single notes (in seconds)
`dur`	vector of duration times of single notes (in seconds)
`vol`	vector of volume of single notes (in seconds)
`freq`	vector of frequencies of notes

Currently only 4 instruments are supported:

1	is a pure sine wave
11	is a simple plucked instrument
21	is a tenor singing ah
22	is a soprano singing ah

There are 3 more commands:
`csndPlayStereo`, `csndCorrphone`, and `csndFourierTones`.

`csndPlayStereo` works very similar to `csndPlayMono`, with the one difference that an additional parameter ranging from -1 to 1 controls the position of the sound source from left to right.

`csndCorrphone` essentially takes 2 frequency vectors and plays the corresponding notes on the left and the right channel. The names indicates a "correlophone", similar to a correlogram. The notes played simultaneously should be very close of the two vectors are highly correlated.

`csndFourierTones` takes a data matrix, interprets each row the coefficients of a Fourier series, and plays tones with these series consecutively.

3 General considerations

Sonification roughly speaking means interpreting numbers as parameters of sound and creating the sounds accordingly. The "natural parameters" for sounds are frequency and volume, and additionally location of the sound source. Using these parameters has the disadvantage that we only can map 3 dimensions of the data into "sound space". Our function `csndFourierTones` can map much higher dimensions, vectors of arbitrary length can be used for the Fourier coefficients. It is known that the human ear can identify overtones up to order 20, therefore we can map data sets of higher dimensions into sounds, at least more easily than into visual representations.

The structure of our sonification model is that we have *instruments* and control their parameters by vectors derived from data. The instruments we are using currently are more or less directly supplied by Csound.

Csound implements the ideas just described as a sort of "assembler language for sound". A typical piece of of code for describing an instrument in Csound looks like this:

```
      instr 11  ; plucked string instrument
idur  =   p3
ivol  =   p4
ifq   =   p5

aout      pluck     ivol, ifq, ifq, 0, 1
          out       aout
      endin
```

This code indicates a call to instrument 11 from the control file (called score file in Csound's terminology) uses parameters 3, 4 and 5 for duration, volume and frequency, and sound is created using the `pluck` opcode (part of the Csound language), which implements a simple plucked instrument. Parameter 1 of any call always is the instrument number, and parameter 2 always is starting time.

Csound has more complicated opcodes for creating sound (our examples use a male and a female singing voice), and it is relatively straightforward to make more of these instruments accessible from R. The canonical reference for Csound, by the way, is [1]. Csound also supports sampling, so we may use any recorded sound (e.g. a lion's roar) and change its parameters (volume, pitch ..) according to data. Csound also supports the Soundfont format used by many popular programs for sample libraries. Many libraries of samples instruments are available, and in particular there are full libraries of all the instruments of GM, the General MIDI standard. Using these libraries we can use sounds of musical instruments (like piano or clarinet) immediately.

Csound is available freely for academic use, and it is multi platform. Currently, implementations for Windows, Macintosh, and many different flavors of UNIX (including Linux) exist. Our Rcsound package requires a running installation of Csound version 4.10 or later.

The interface mechanism of our package to call Csound is very simplistic. The package writes 2 temporary files and then calls Csound with these files as input input files. This kind of simplicity has the advantage that it works without problems on any of the supported platforms. It is tested on Windows, Linux, and Solaris, but should easily also run on all the other platforms supported by R and Csound.

4 Planned further development

As stated already out package has to be understood as a working prototype for using R as a test bed for sonification. Sonification is a relatively new area, and no standard representation types have been established so far. [2] gives an interesting nontrivial application, and also has a nice list of references for recent developments.

Given this experimental character of an emerging field, it seems important to embed sound creation directly into a statistics package. Having R as a host environment for sonification using Csound combines one of the most powerful and developer friendly statistics packages with one of the most powerful sound creation packages.

The next step is needed is to do research about "sonification that works", i.e. instruments or sound creating methods which make it easy to recognize patterns in data by listening. Hopefully, our package can make the task ahead more easily manageable and allows us to create data aware instruments. Using Csound hopefully offers a much wider range of acoustical options than translating data into simple MIDI messages.

References

[1] Richard Boulanger. *The Csound Book*. MIT Press, 2000.

[2] T. Hansen, M. H. Hansen, and H. Ritter. Sonification of markov chain monte carlo simulations. In *Proceedings of the 2001 International Conference on Auditory Display*, 2001.