Olivier Lartillot, Academy of Finland research fellow, 2009-2014, Finnish Centre for Interdisciplinary Music Research, U. Jyväskylä

MiningSuite

code.google.com/p/miningsuite

SIGMINR
signal processing

"*MIRtoolbox 2.0*": User-friendly but powerful operators

```
a = sig.input('myfile','Extract',0,60,'s');
b = sig.spectrum(a,'dB');
c = aud.brightness(b)
sig.ans
```

Using semi-colons, design a complete dataflow while postponing the computation.

a, b and c are sig.design objects: they store the whole dataflow design, which is evaluated only when you want to display the results.
Operators' main routine simply calls sig.operate, which performs the techniques underlying this framework (such as efficient memory management) in a transparent way.

sig.data adds a syntactic layer on top of *Matlab* that makes operators' code simpler.

Whereas in standard *Matlab* code, the processing of matrix dimensions makes the code somewhat obscure, here x.**size('sample')** gives the number of samples of sig.data x, x.**times**(y) multiplies 2 sig.data objects, respecting dimension type congruency. x.**apply**(@xcorr,{},{'sample'},2) applies *xcorr* along the sample dimension. The last argument notifies that *xcorr* does not work for matrices with more than 2 dimensions. The extra dimensions are automatically covered via loops.

Resulting signal data are sig.signal objects, encapsulating all relevant information (Exported data includes details of operations & settings), and providing standard post-processing operations.

Information such as time positions are regenerated on the fly, using sig.axis.

sig.input sig.envelope sig.spectrum sig.flux sig.filterbank sig.hist sig.stat sig.centroid sig.skewness
sig.play sig.rms sig.cepstrum sig.specslope sig.cluster sig.dist sig.mean sig.spread sig.median sig.kurtosis
sig.save sig.zerocross sig.rolloff sig.peak sig.simatrix sig.std sig.flatness

AUDMINR
auditory modelling

aud.spectrum aud.roughness aud.pitch aud.dist aud.event
aud.mfcc aud.hnr aud.brightness aud.flux aud.attack aud.release
aud.inharmonicity aud.cluster aud.simatrix aud.eventdensity
aud.irregularity aud.filterbank aud.envelope aud.segment

VOCMINR
voice analysis

voc.minr voc.pitch voc.loudness voc.alpha voc.voice
voc.jitter voc.shimmer voc.hammarberg
voc.harmonics voc.formant voc.pause

SEQMINR
sequence processing

seq.sequence: Routines for symbolic sequence management. Elements of sequences, of class seq.event, can be subsequences themselves. seq.param manages the underlying set of parameters, which can be hierarchical (with one parameter type more general than another one). *common(p1,p2)* returns the common parametric description. *p1.implies(p2)* tests whether *p2* is more general than *p1*.
Succession of elements are linked with seq.syntagm object, with parametric description automatically computed.

PATMINR
pattern mining

Detects all repetitions of sequential *patterns* in symbolic sequences along all parametric dimensions altogether.

applied in music to metrical and motivic analysis (cf. below).

Models successive repetitions as *cycles*. Constructs a pattern dictionary (as a tree) and builds pattern *occurrences* on the symbolic sequence.

MIDI sequences

transcription from audio

MUSMINR
music analysis

Integrates both audio-based and symbolic-based methods.

mus.spectrum mus.beatspectrum
mus.keyspectrum mus.pulseclarity
mus.keysom
mus.tonalcentroid mus.hcdf mus.metroid
mus.keyclarity mus.majorness

Signal and statistical methods can also be applied to symbolic data (onset curve, key profile...)

mus.sequence

mus.param:
pitch: chromatic: MIDI pitch
diatonic: pitch spelling
time: onset-offset time in s.
metrical position

mus.key
Detection of keys and modes based on comparison of pitches with templates. Impacts pitch spelling.

mus.tempo **mus.metre**
Construction of metrical structure through cyclical pattern mining using PatMinr.

Further musicological analyses: motivic analysis (using PatMinr), ornamentation reduction, segmentation, ...

Important parameters are actually defined between successive notes, i.e. seq.syntagm objects:
pitch interval, rhythmic value (quantised)
Because all dimensions are interconnected, the whole set of analyses is performed for each successive note, using mus.minr.

The *MiningSuite* is significantly optimised (in speed and memory) compared to *MIRtoolbox*.
Fully rewritten using recent *Matlab* object-oriented programming capabilities