MIRtoolbox: Sound and music analysis of audio recordings using Matlab

MUS4831, Olivier Lartillot

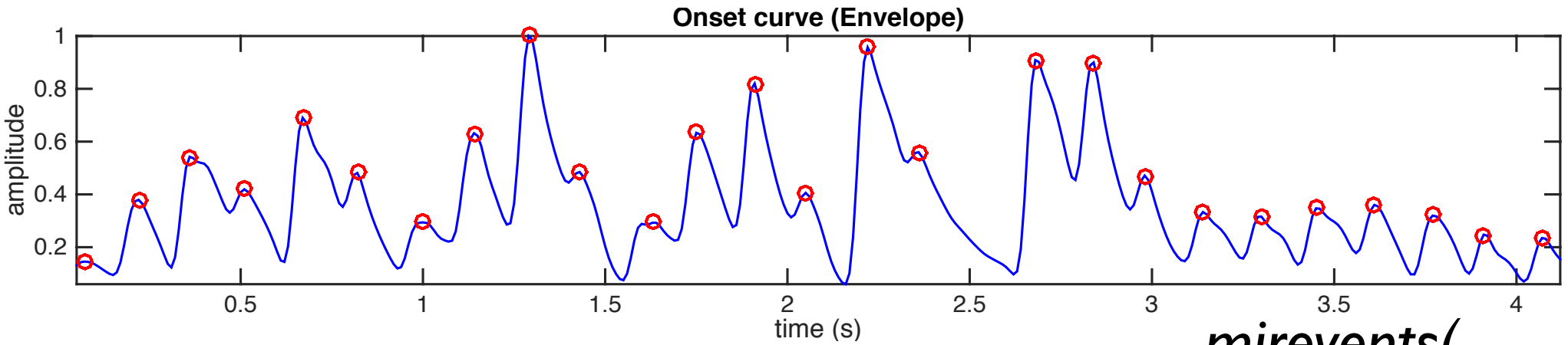
Part II, 9.11.2017

Part 2

- Rhythm, metrical structure
- Tonal analysis
- Segmentation, structure
- Statistics
- Music & emotion
- Advanced use

*mir*events

event detection function

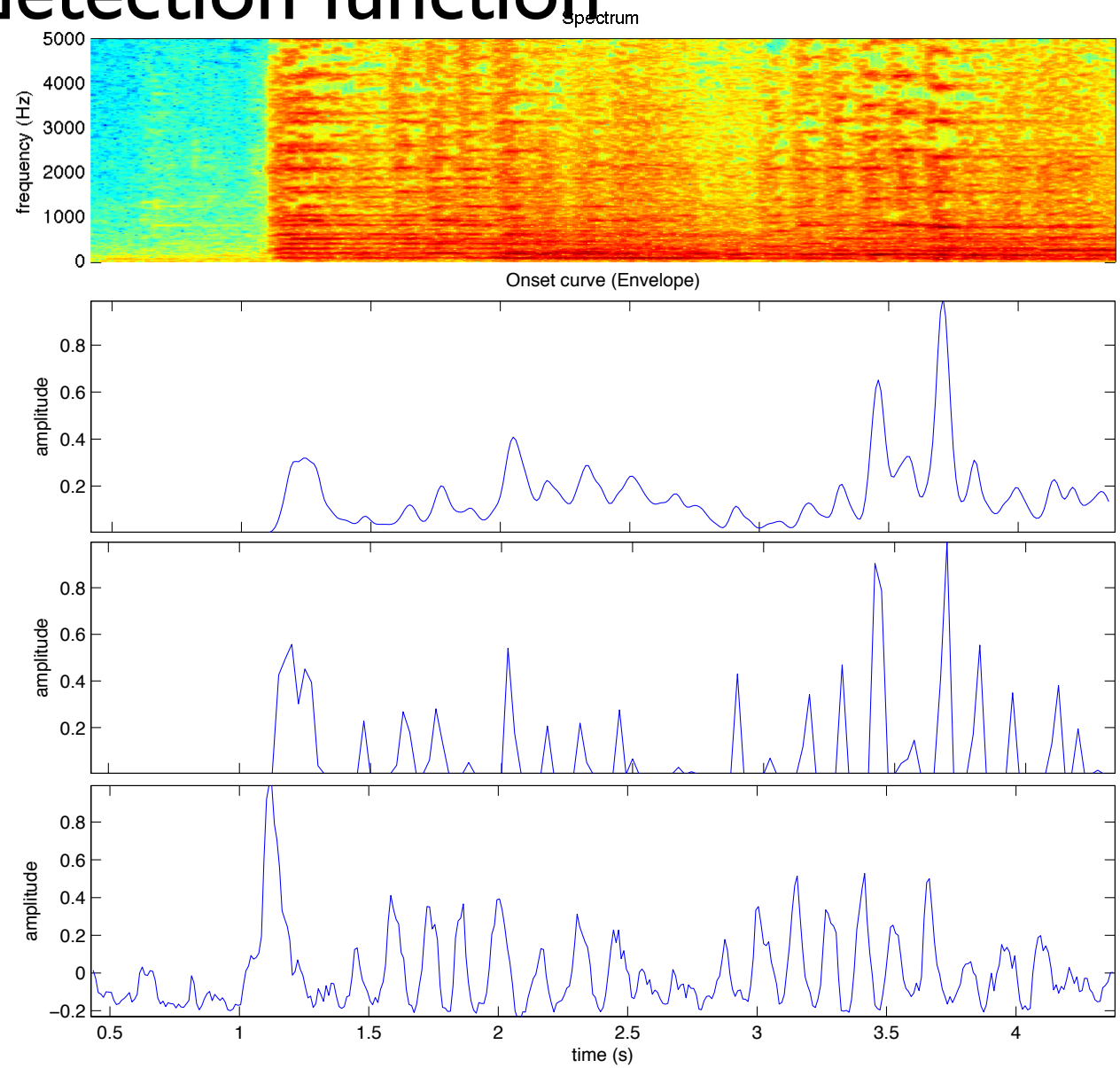


*mirevents(…, **'Contrast'**, …)*

- *mirevents('ragtime.wav')*
  - *mirpeaks(mirsum(mirspectrum(…,'Frame')))*
- *mirevents('ragtime.wav', **'Filter'**)*
  - *mirpeaks(mirsum(mirenvelope(mirfilterbank(…,'NbChannels', 40))))*
- *mirevents('ragtime.wav', **'SpectralFlux'**)*
  - *mirpeaks(mirflux(…,'Inc','Halfwave'))*

*mir*events

event detection function

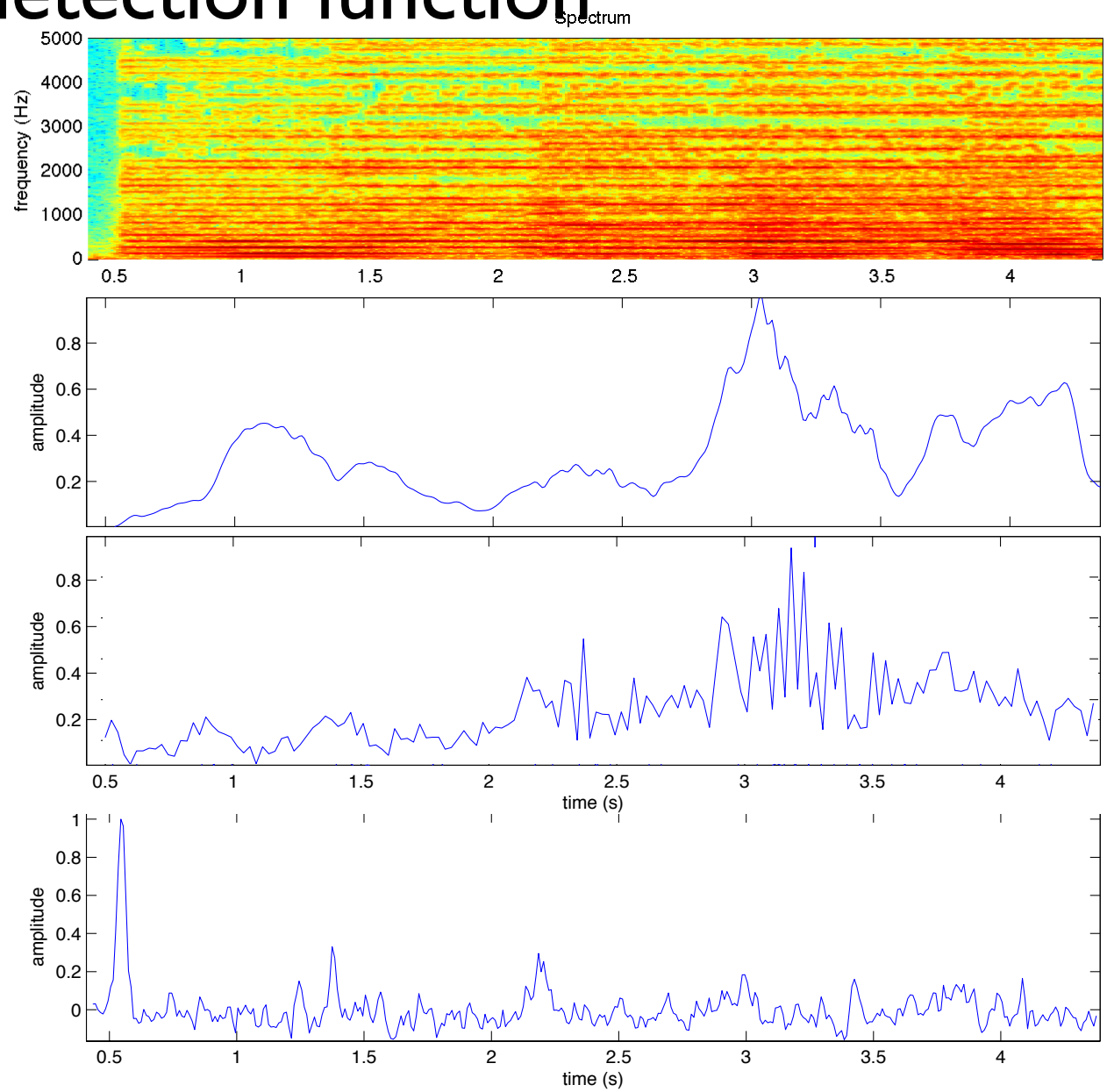


- *'Envelope', 'Filter'*: changes in dynamics
- *'SpectralFlux'*: global spectral changes
- ***'Emerge'***: local changes in particular frequency regions

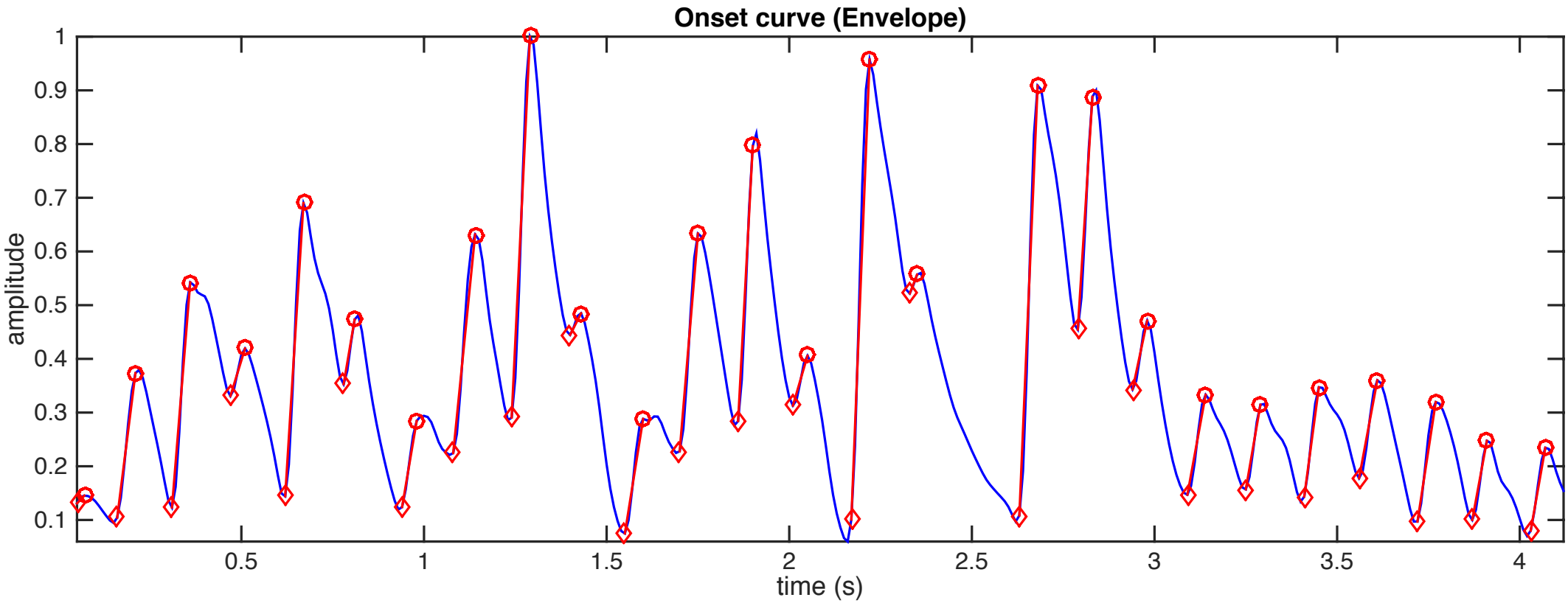
*mir*events

event detection function

- *'Envelope', 'Filter'*: changes in dynamics
- *'SpectralFlux'*: global spectral changes
- ***'Emerge'***: local changes in particular frequency regions

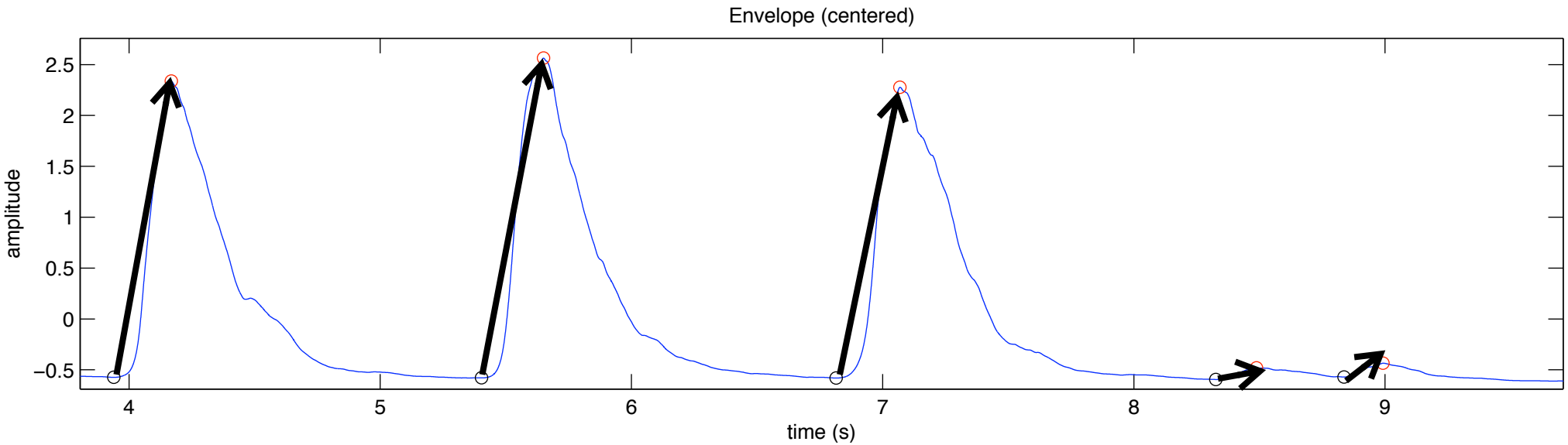


mirevents(..., 'Attack')



mirattackslope

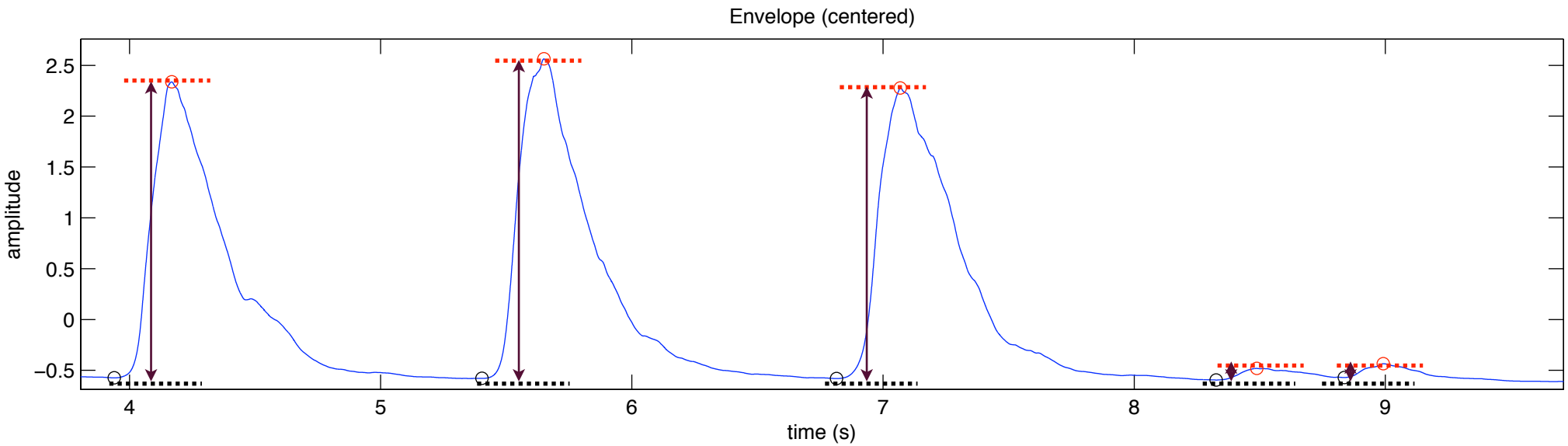
average slope of note attacks



- *o = mirevents('ragtime.wav','Attacks')*
- *mirattackslope(o)*

mirattackleap

amplitude of note attacks



- *o = mirevents('ragtime.wav','Attacks')*
- *mirattackleap(o)*

Tempo estimation?

- '*george.wav*'
- What tempo in BPM? You can tap on the beat while listening:
  - http://www.all8.com/tools/bpm.htm
- How to estimate tempo using the *MIRtoolbox* operators presented last week?

*mir**tempo***

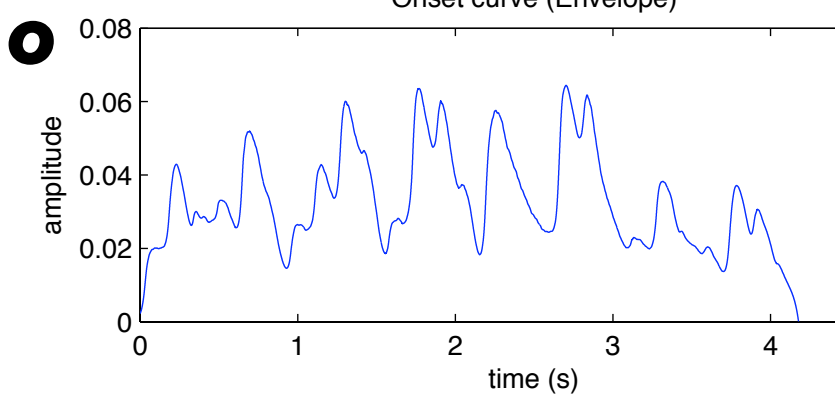
tempo (in beats per minute)

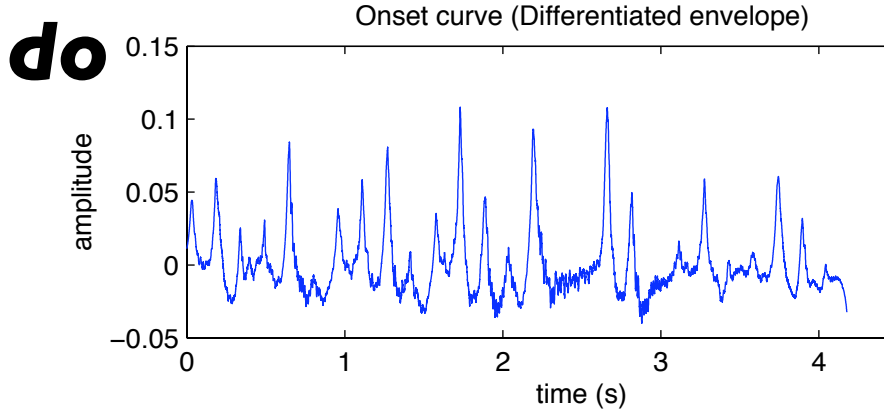
Roughly:

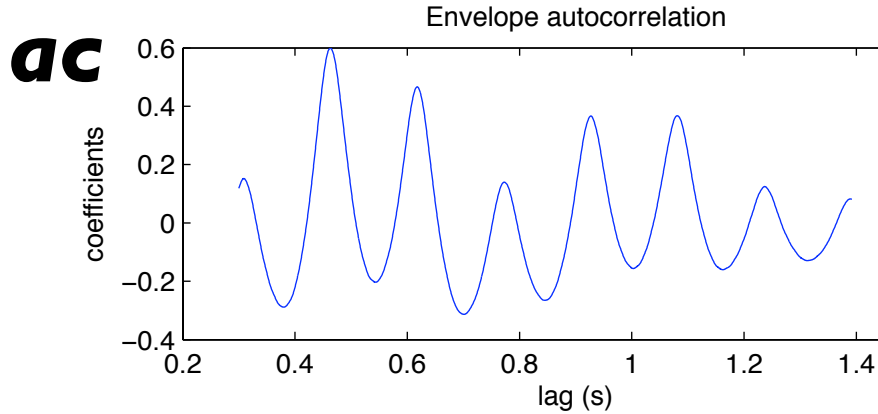
- ***o*** = *mirevents(‘file.wav’, ‘Filter’)*
- ***do*** = *mirevents(**o**, ‘Diff’)*
- ***ac*** = *mirautocor(**do**, ‘Resonance’)*
- ***pa*** = *mirpeaks(**ac**, ’Total’, 1)*

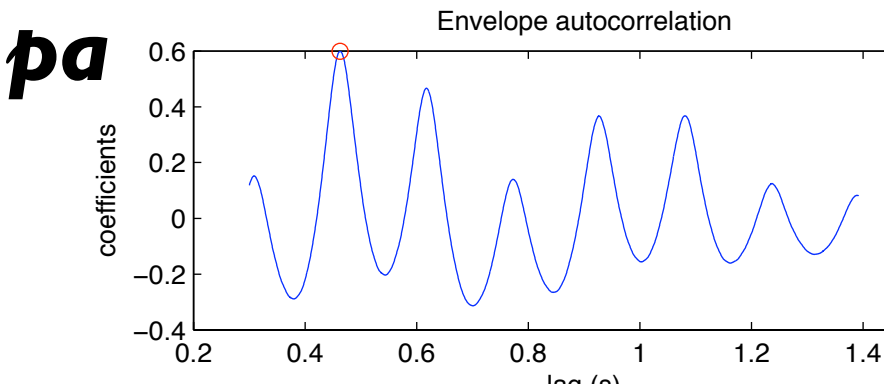
In short:

- *[**t**, **pa**] = mirtempo(‘file.wav’)*





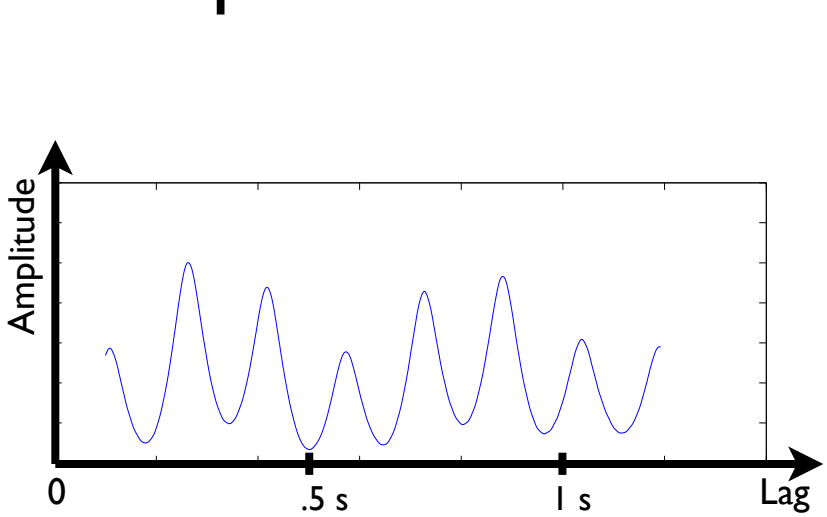




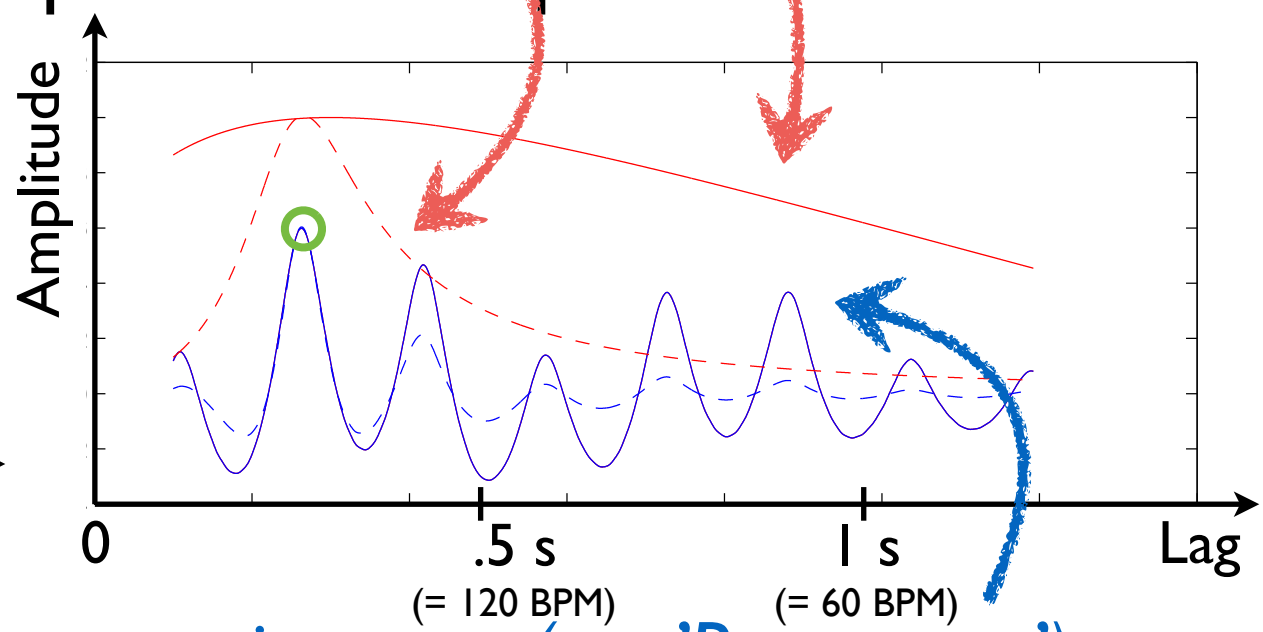
mirtempo

resonance curves

- *mirautocor(…, 'Resonance', 'Toiviainen')* *(Toiviainen & Snyder, 2003)*
- *mirautocor(…, 'Resonance', 'vanNoorden')* *(van Noorden & Moelants, 2001)*
- Emphasis on the best perceived tempi



mirautocor



mirautocor(…, 'Resonance')

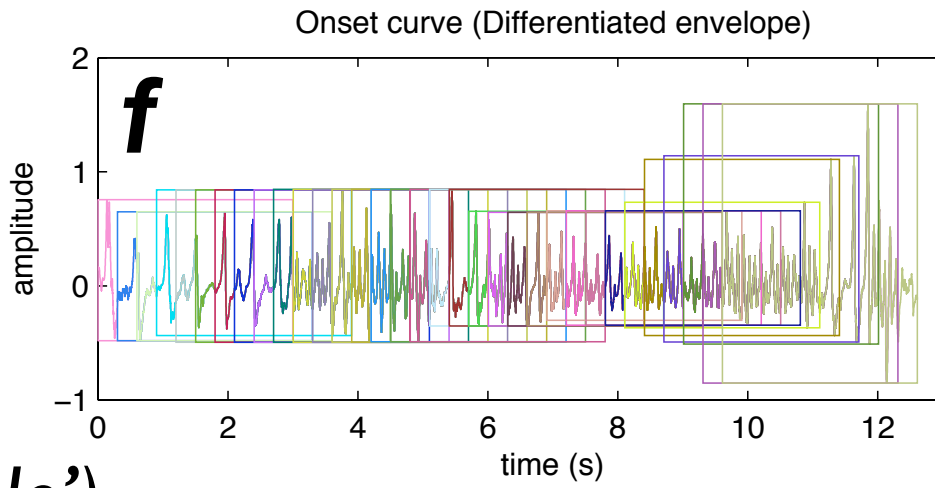
*mir**tempo***

tempo estimation

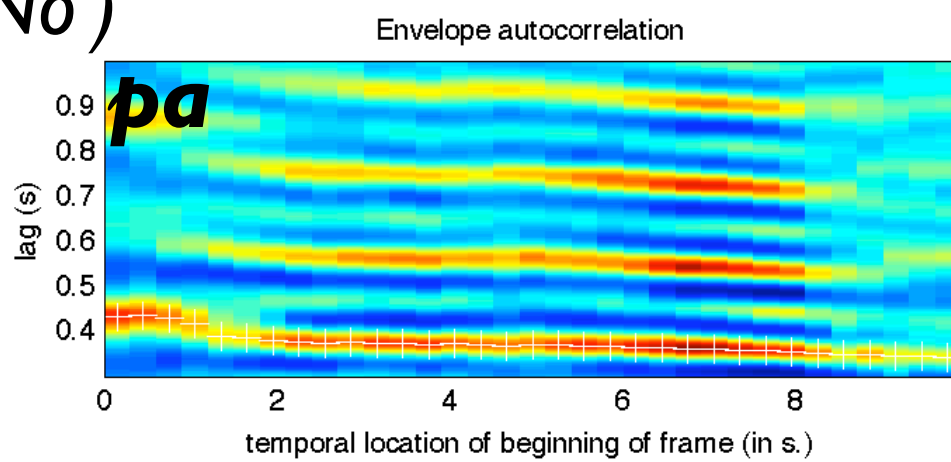
- How to estimate tempo for such audio excerpt?
  - *'czardas.wav'*

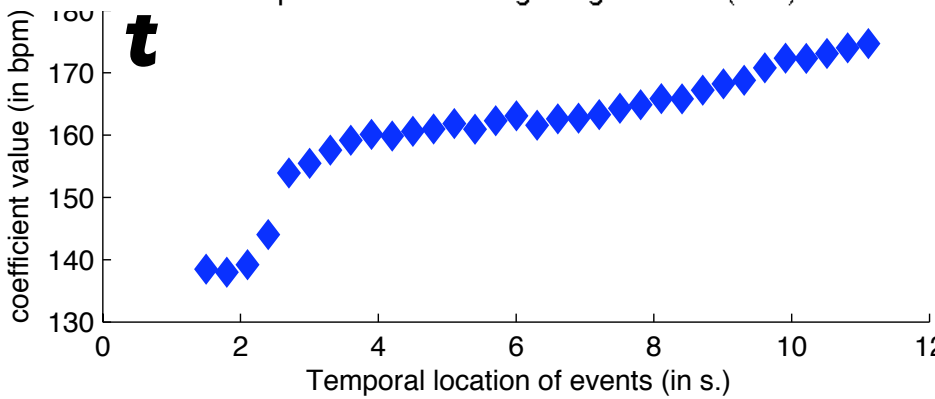
*mir*tempo

tempo (temporal evolution)



- **o** = *mirevents('mysong', 'Detect', 'No')*
- **do** = *mirevents(o, 'Diff')*
- **f** = *mirframe(do)*
- **ac** = *mirautocor(f, 'Resonance')*
- **pa** = *mirpeaks(ac, 'Total', 1)*



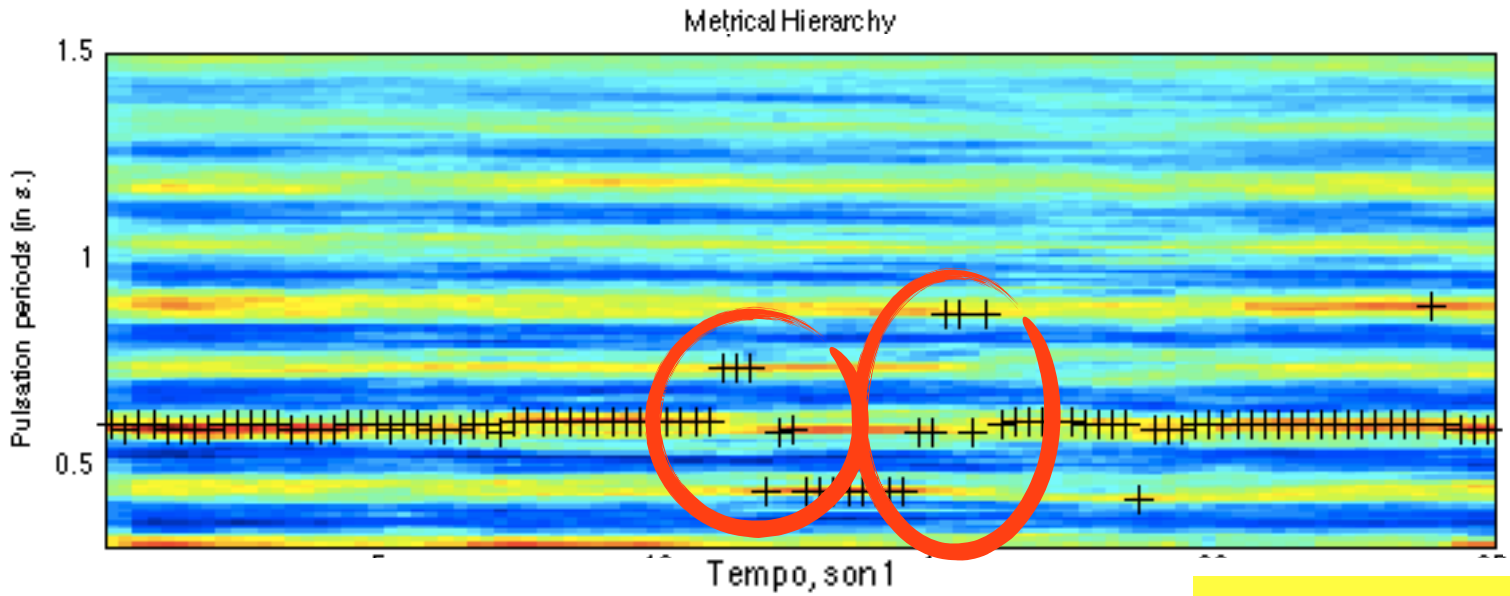


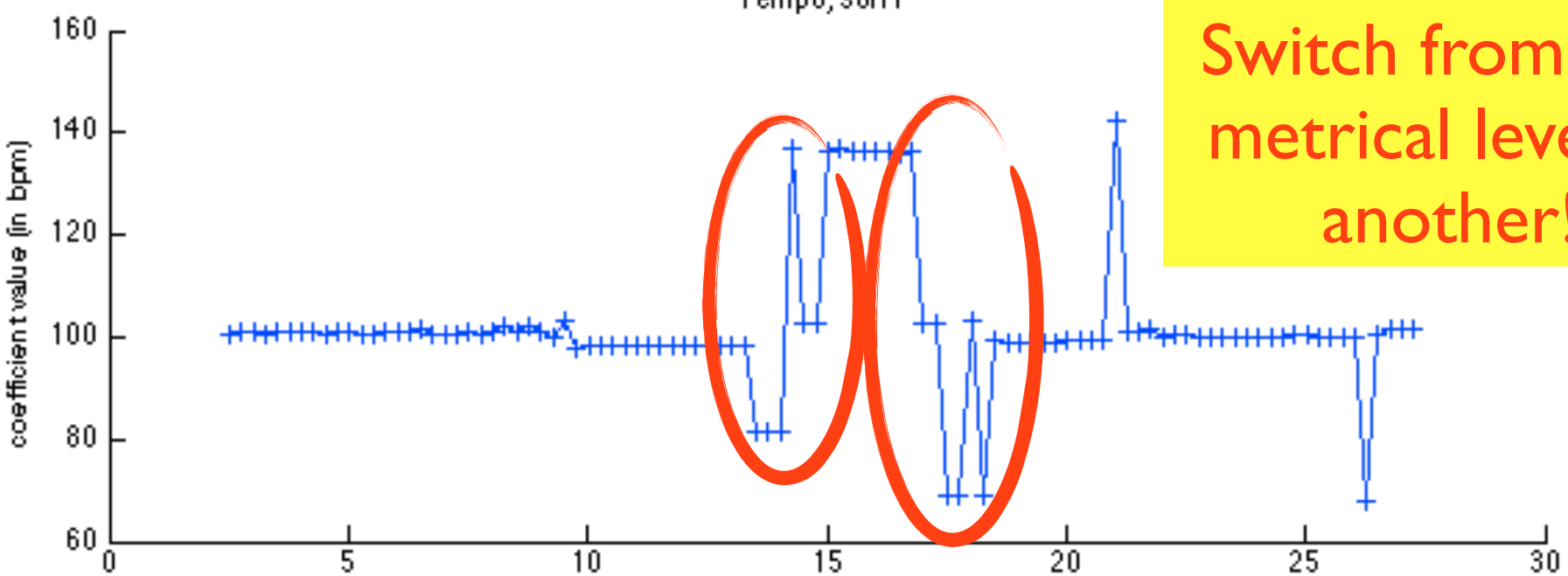
In short:

- *[**t**, **pa**] = mirtempo('mysong', 'Frame')*

Try for instance: mirtempo('george.wav', 'Frame')

*mir*tempo





Switch from one metrical level to another!

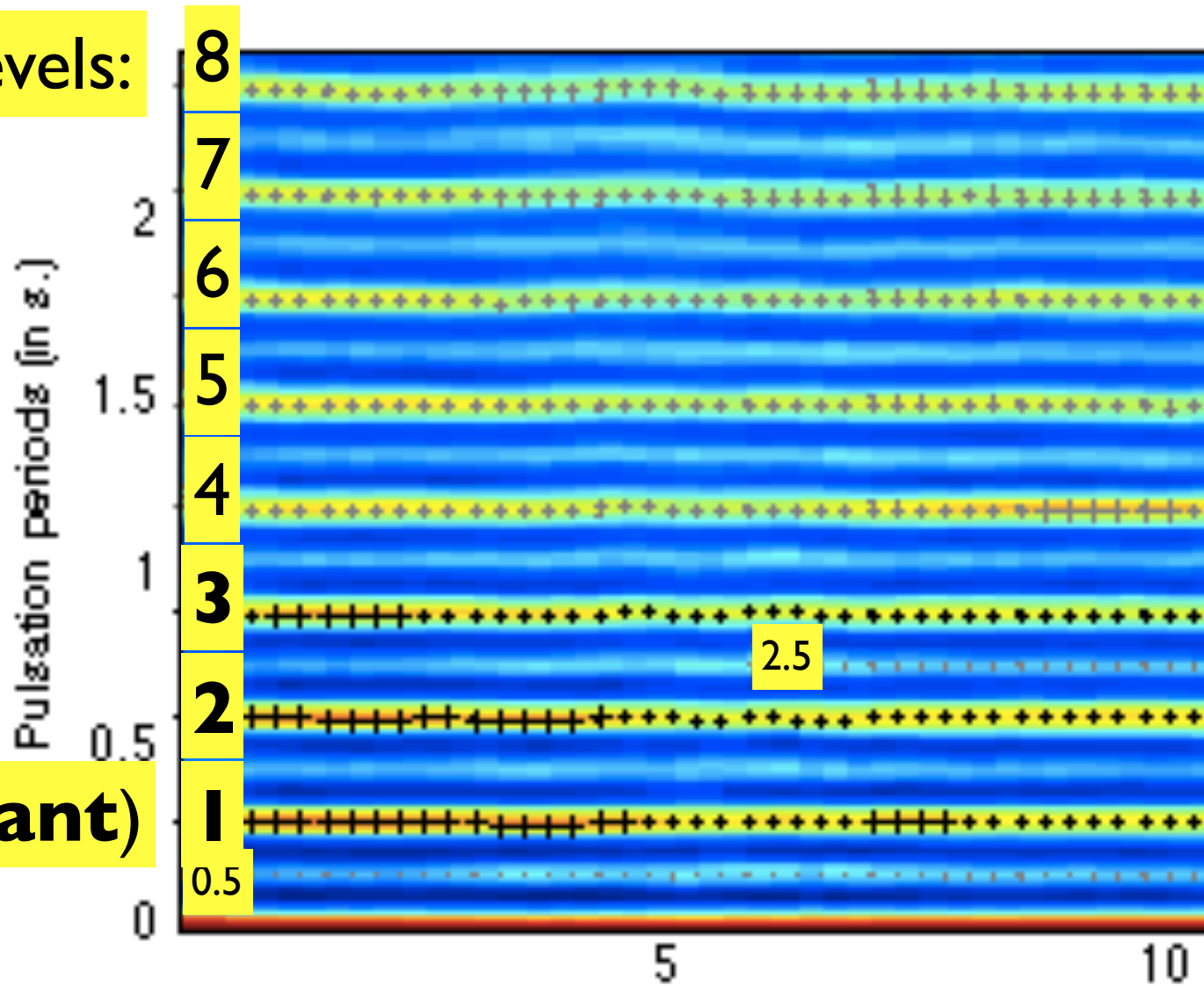
Try for instance: mirmetre('george.wav')

mirmetre

tracking all metrical levels

Metrical levels:

- Level *N*'s period is *N* times level 1's period.



(**dominant**)

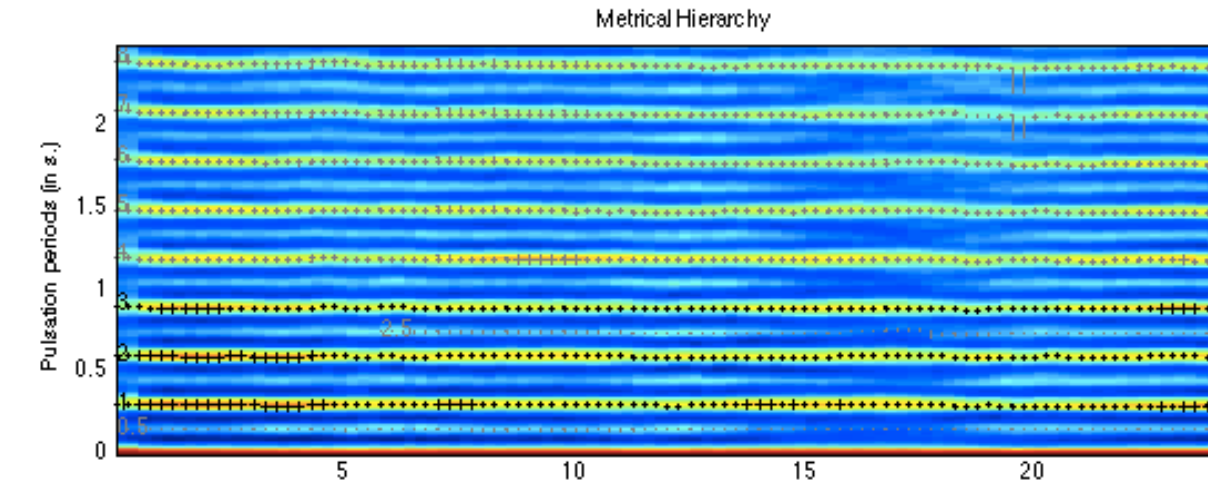
mirmetre

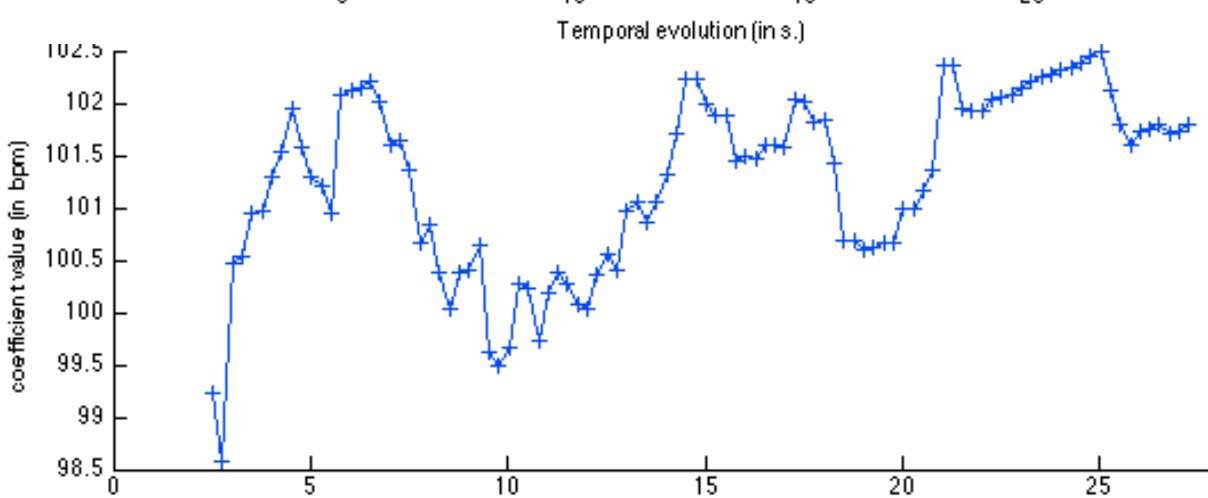
tracking all metrical levels

- *m = mirmetre(…)*
- *t = mirtempo(m)*

or:

- *[t m] = mirtempo(…, 'Metre')*





- The tempo curve is associated to one particular metrical level.

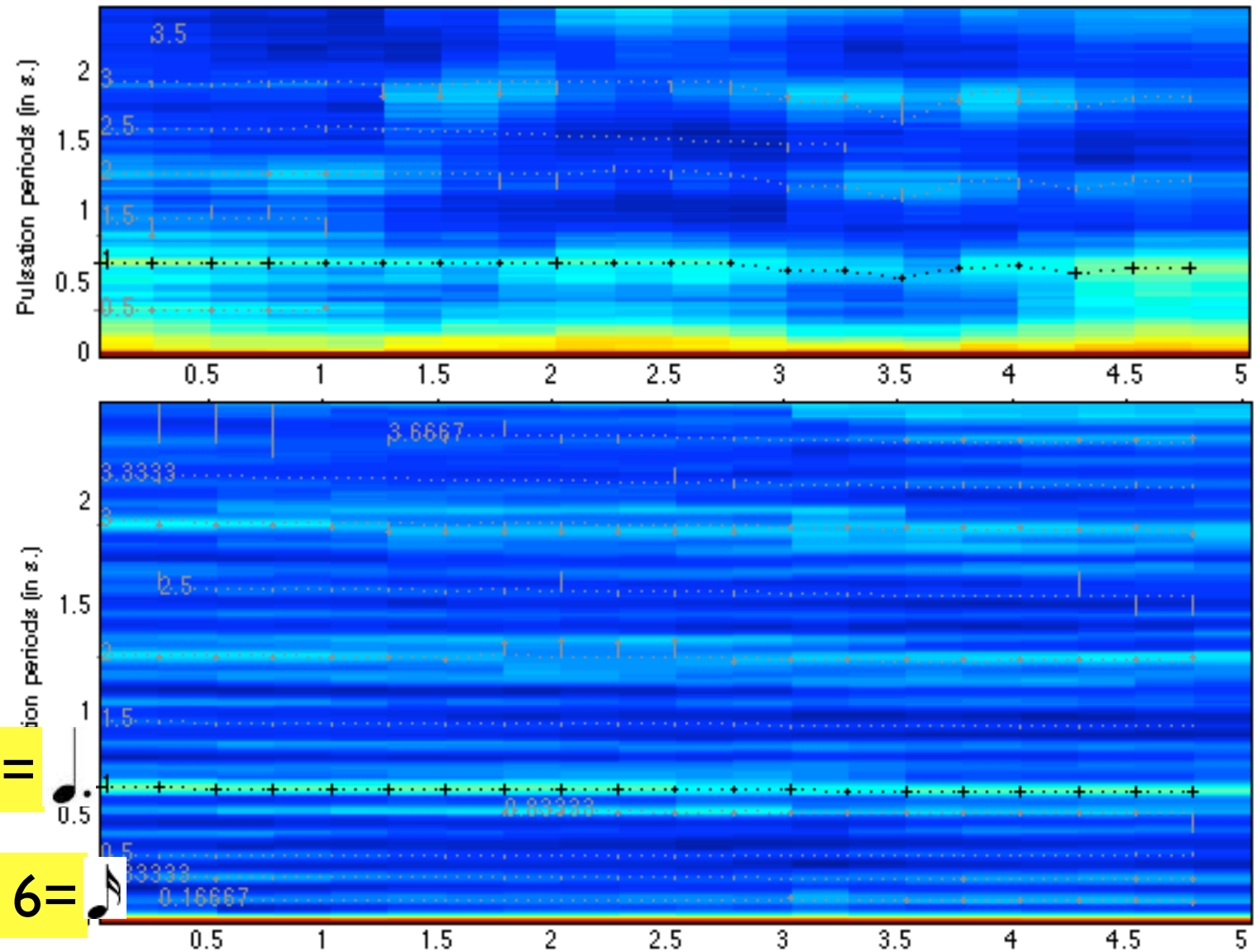
Influence of the onset detection method

mirmetre

tracking all metrical levels

‘Spectral Flux’

‘Emerge’



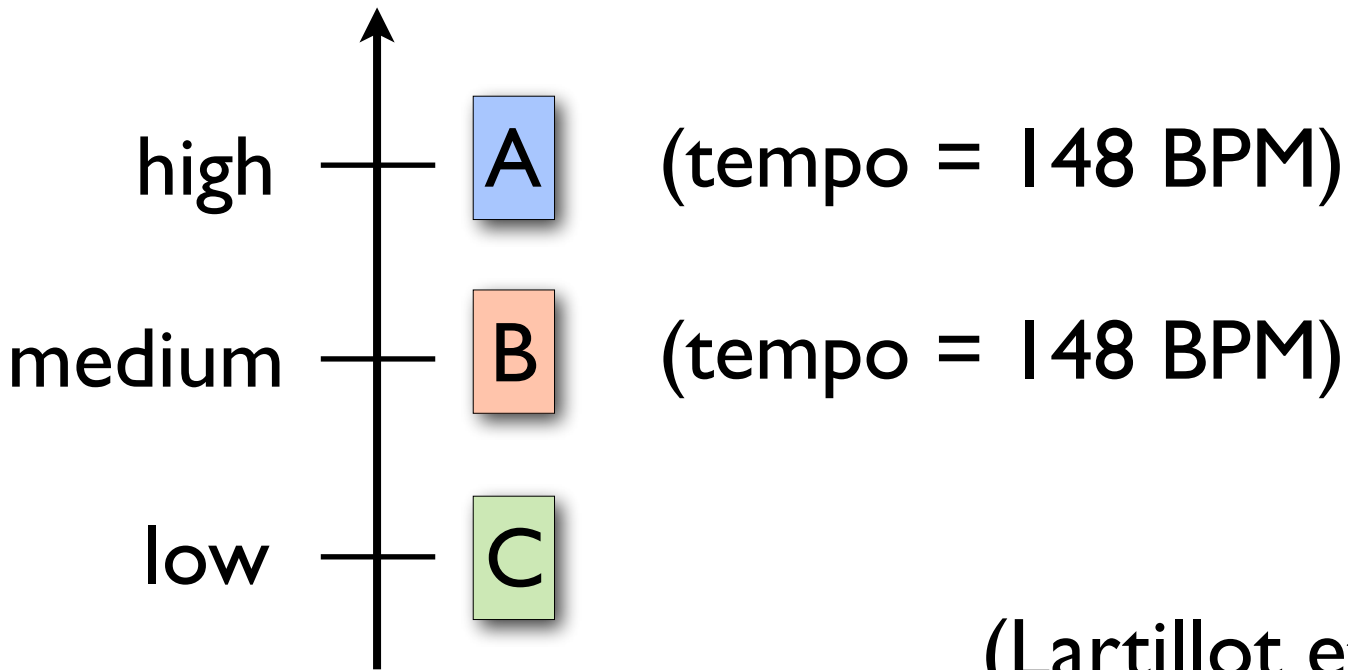
M. Bruch, *Violin Concerto No.1 in G minor*, op.26, *Finale* (*Allegro energico*)

1= ♩.

.16= 𝅘𝅥𝅯

Beat/rhythmic/metrical strength, clarity

- Subjective judgment:
  - How easily I can perceive the underlying pulsation in music.

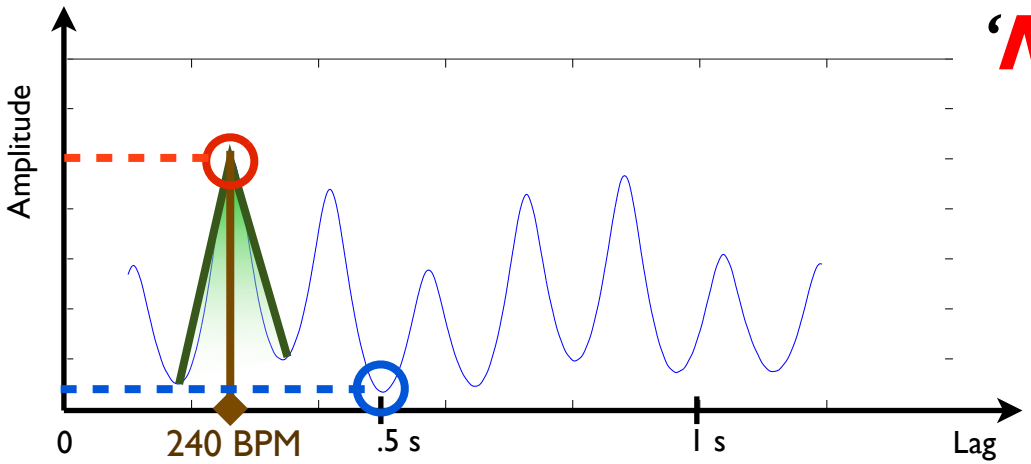


(Lartillot et al., 2008)

mirpulseclarity

beat strength

- Characterisation of the autocorrelation function



'MaxAutocor' **'MinAutocor'**

'KurtosisAutocor'

'TempoAutocor'

'EntropyAutocor'

0

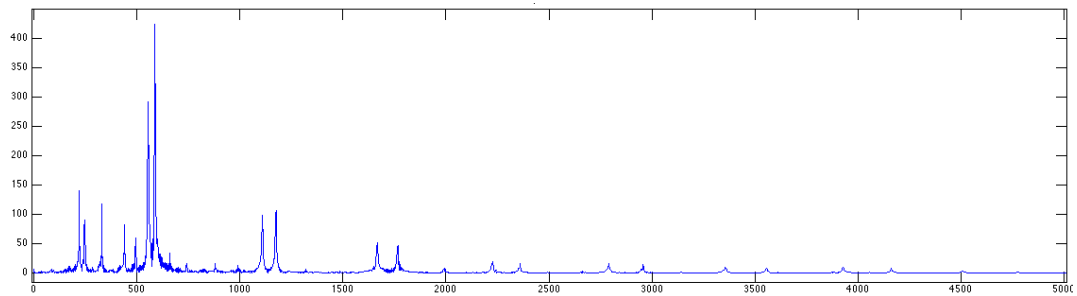
(Lartillot et al., 2008)

- cf. also *beat strength* (Tzanetakis, Essl & Cook, 2002): variability of the autocorrelation curve throughout time

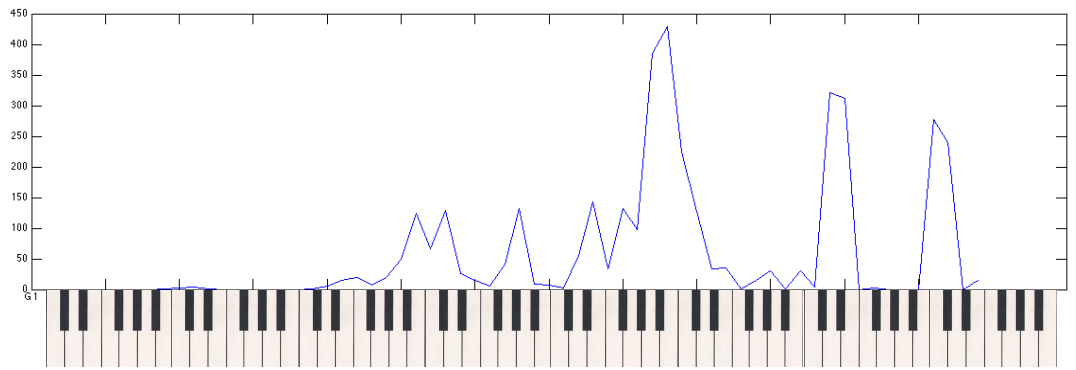
mirchromagram

energy distribution along pitches

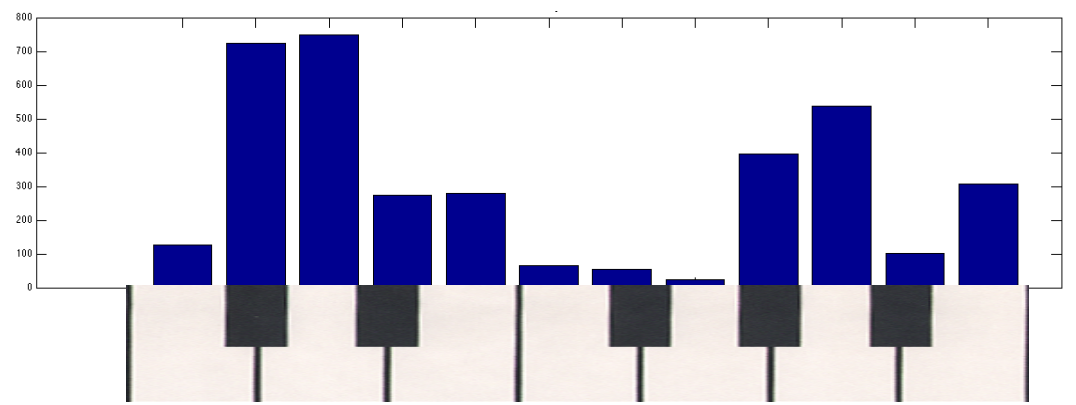
- ***s*** = *mirspectrum(**a**)*



- ***c*** = *mirchromagram(**s**, 'Wrap', 'no')*



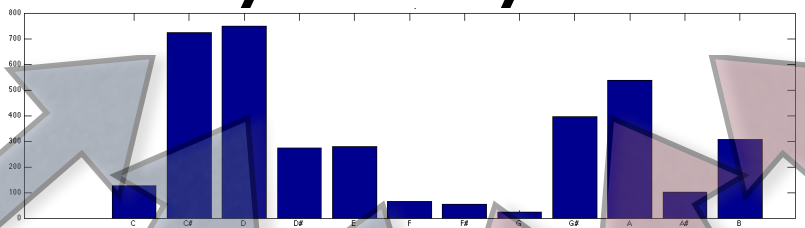
- ***c*** = *mirchromagram(**c**, 'Wrap', 'yes')*



*mir*keystrength

probability of key candidates

mirchromagram

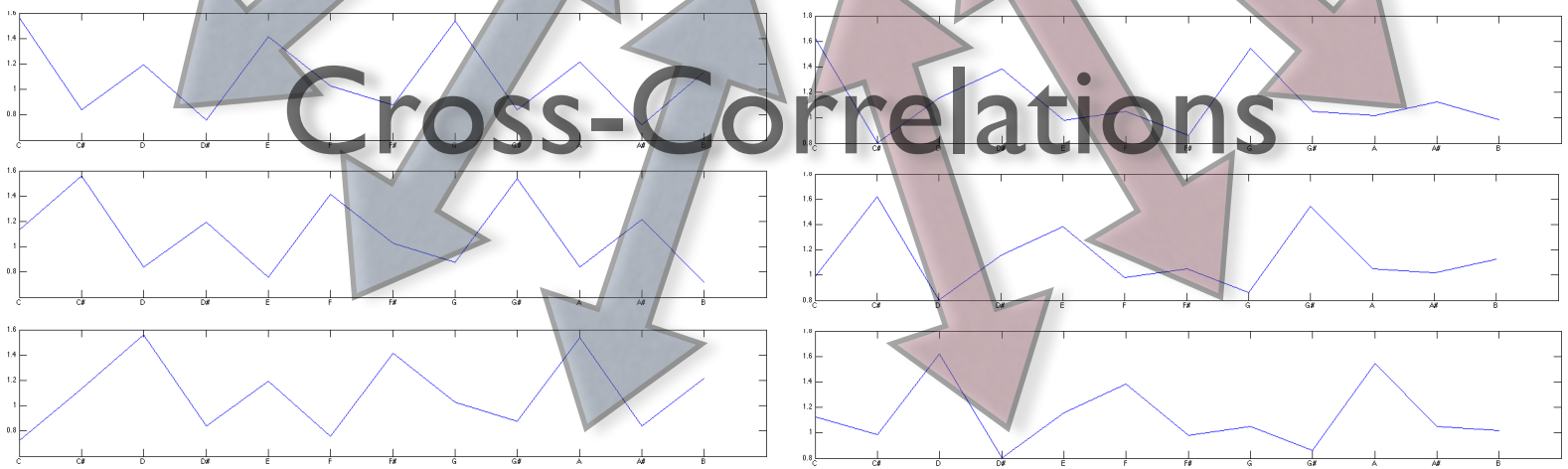


C major

C# major

D major

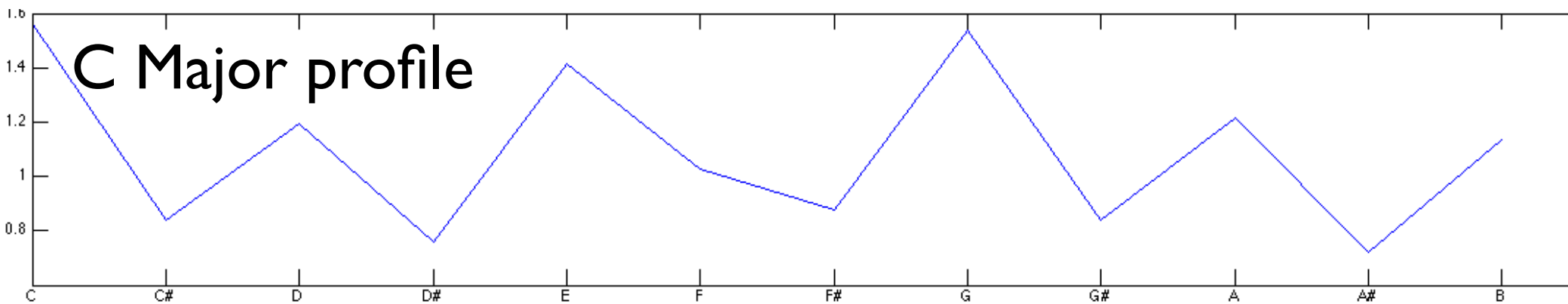
Cross-Correlations



C minor

C# minor

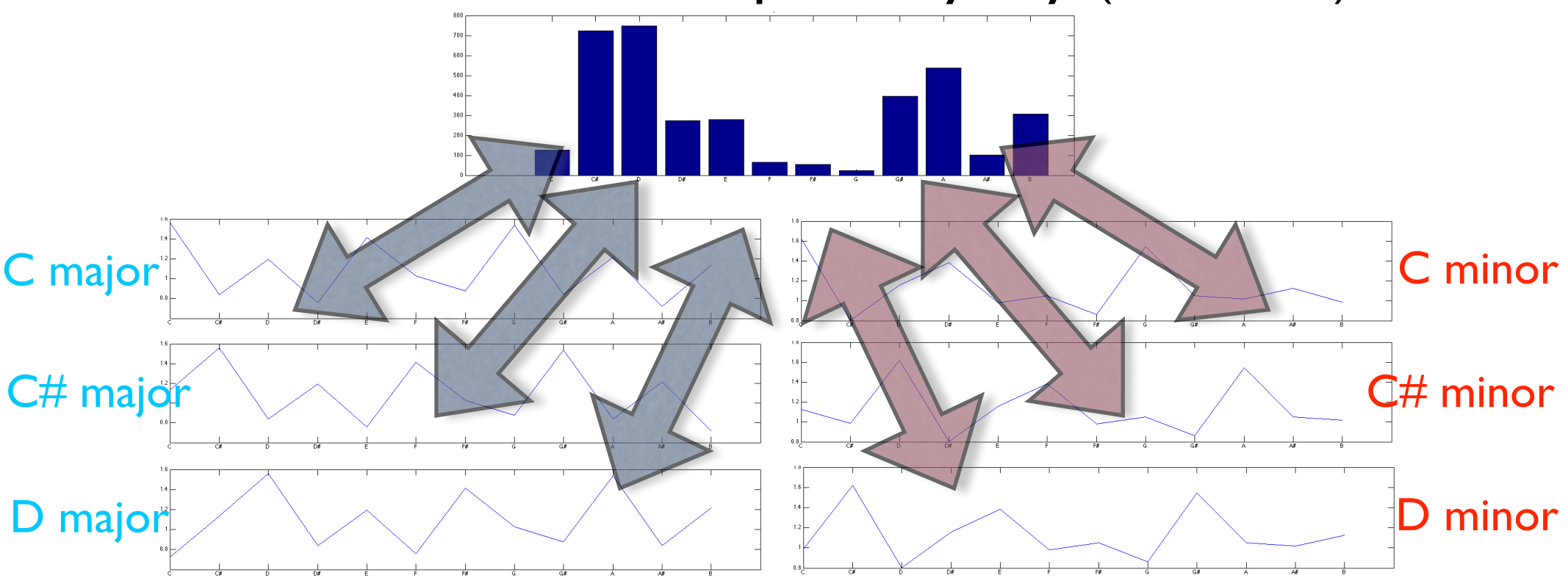
D minor



*mir*keystrength

probability of key candidates

- Chromagram compared to typical chromagrams representing each possible key (or mode).
- Detection of the most probably key (or mode).



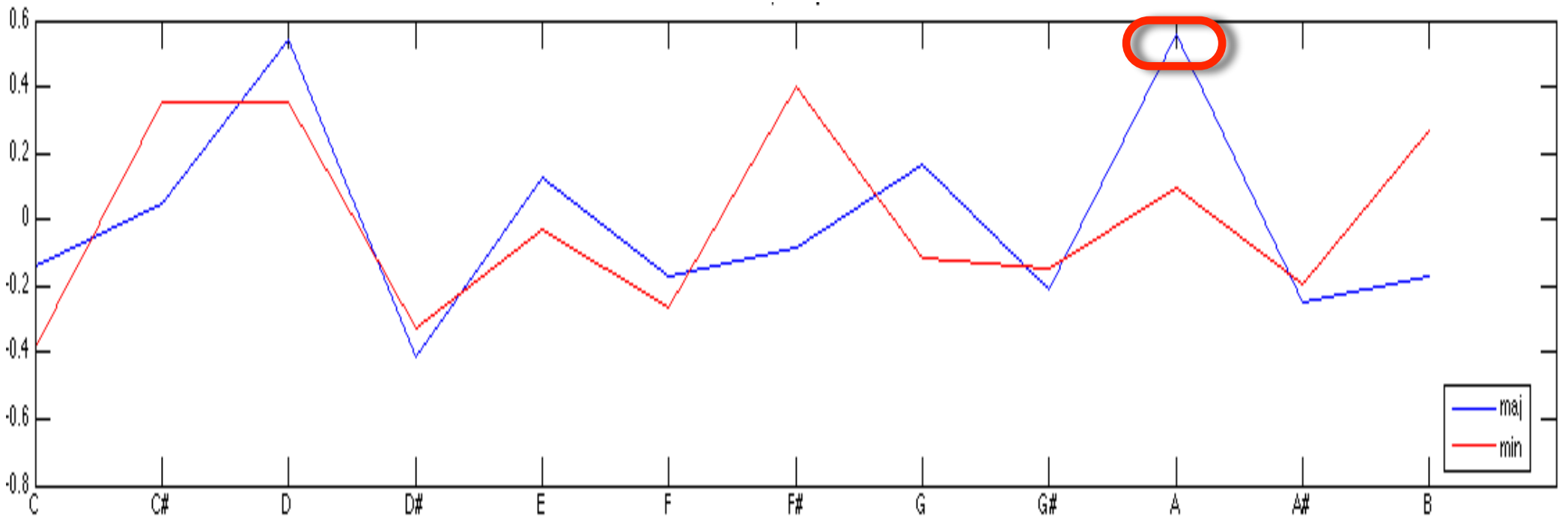
Krumhansl, *Cognitive foundations of musical pitch*. Oxford UP, 1990. Gomez, "Tonal description of polyphonic audio for music content processing," *INFORMS Journal on Computing*, 18-3, pp. 294–304, 2006.

*mir*key

tonality estimation

mirpeaks(mirkeystrength(...))

*mir*keystrength

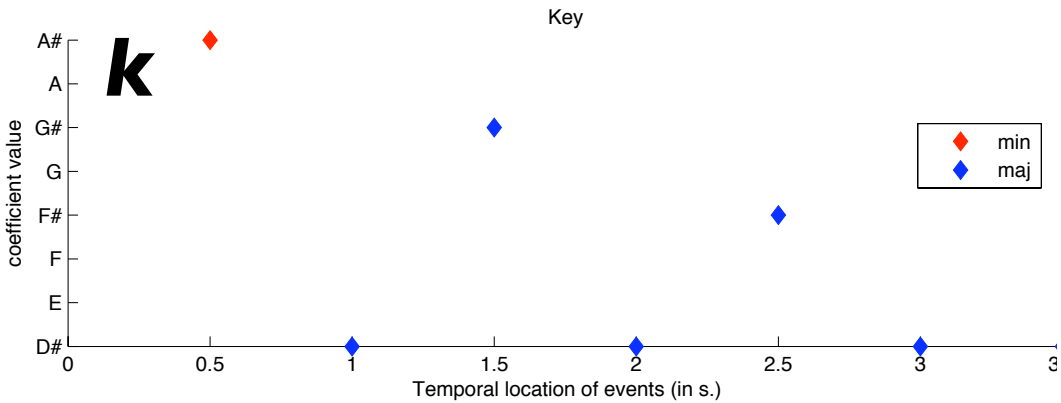


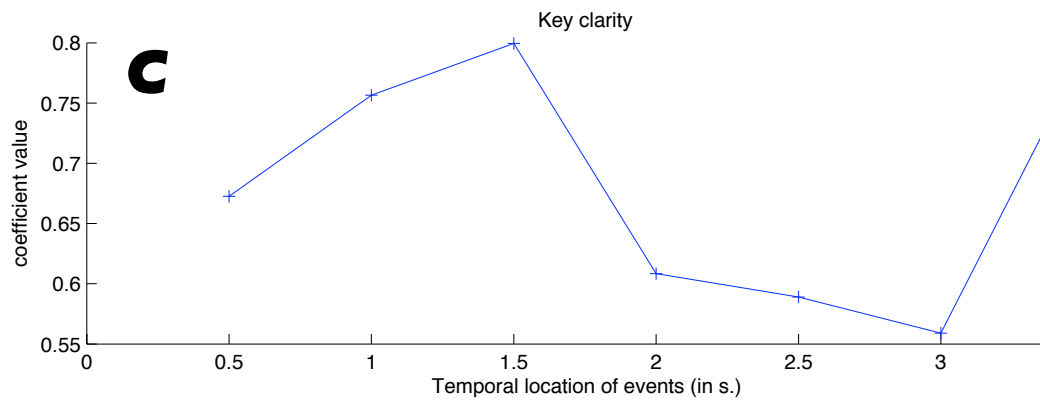
- *[**k c s**] = mirkey(...)*

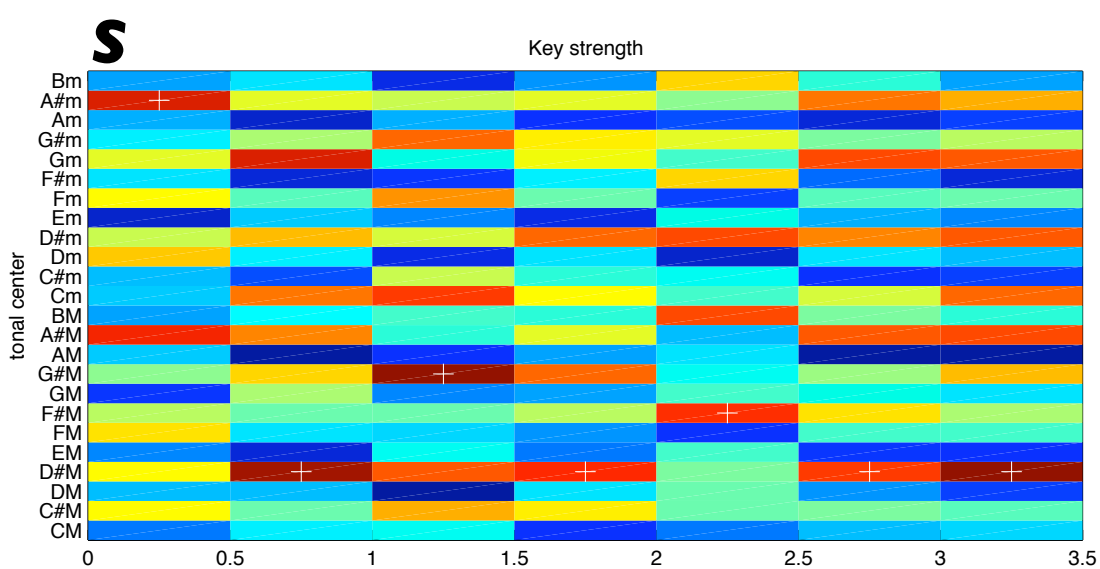
*mir*key

tonality estimation

- *[**k c s**] = mirkey(..., 'Frame')*

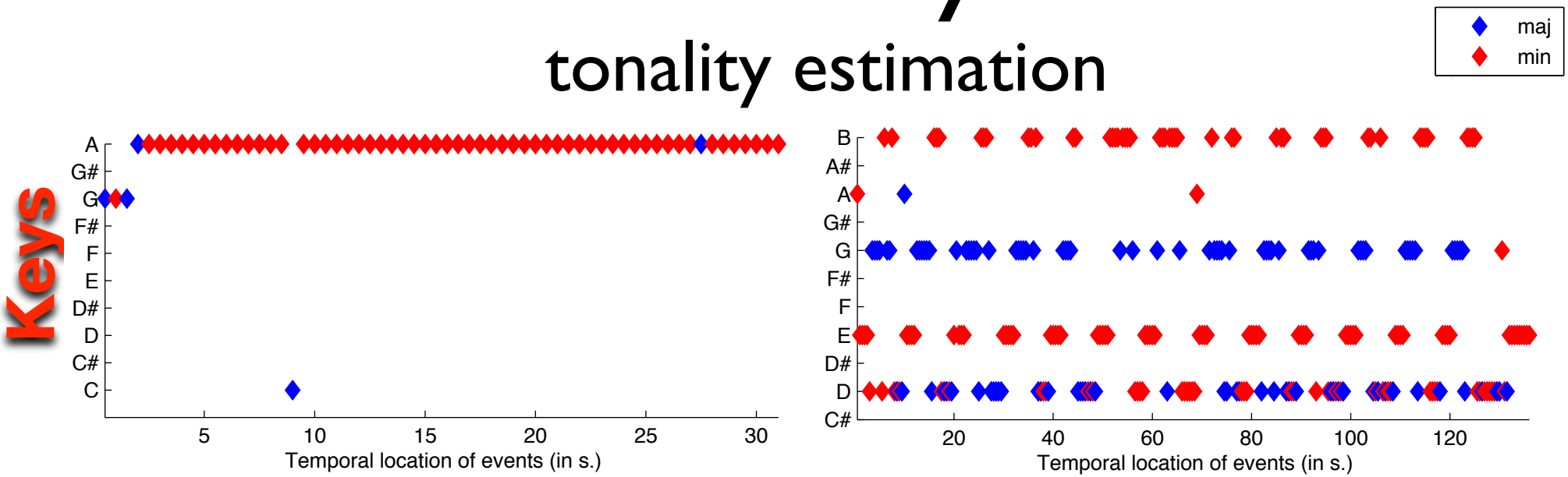






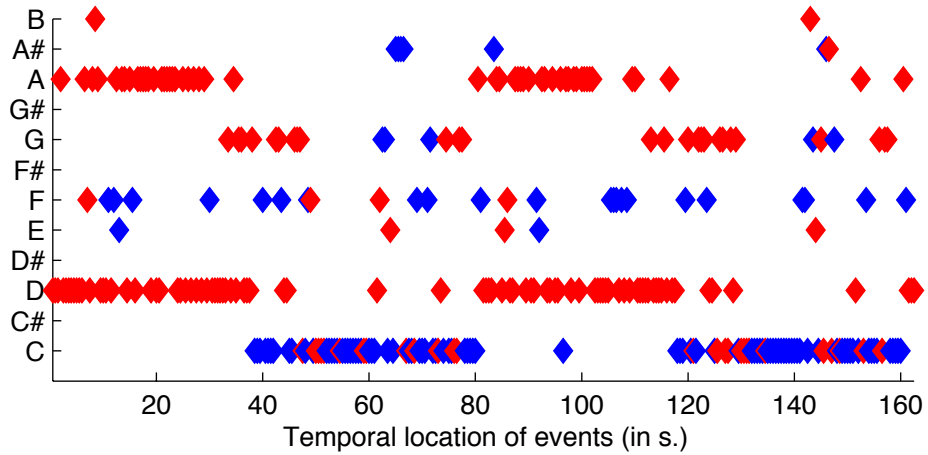
mirkey

tonality estimation

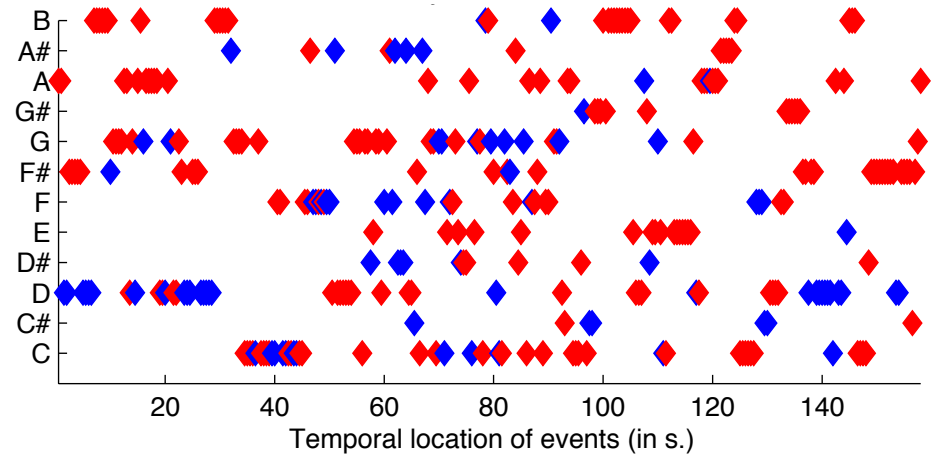


monteverdi.wav

tiersen.wav



beethoven9.wav

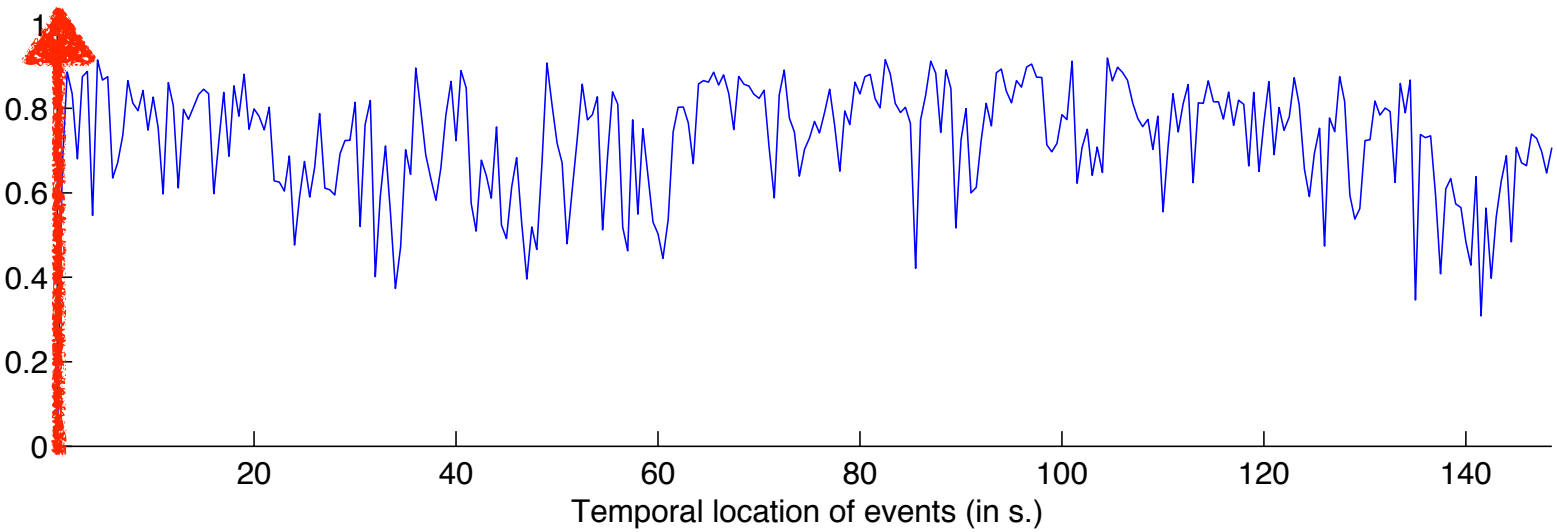


schoenberg.wav

- *[k **c** s]* = *mirkey(..., 'Frame')*

clear tonality

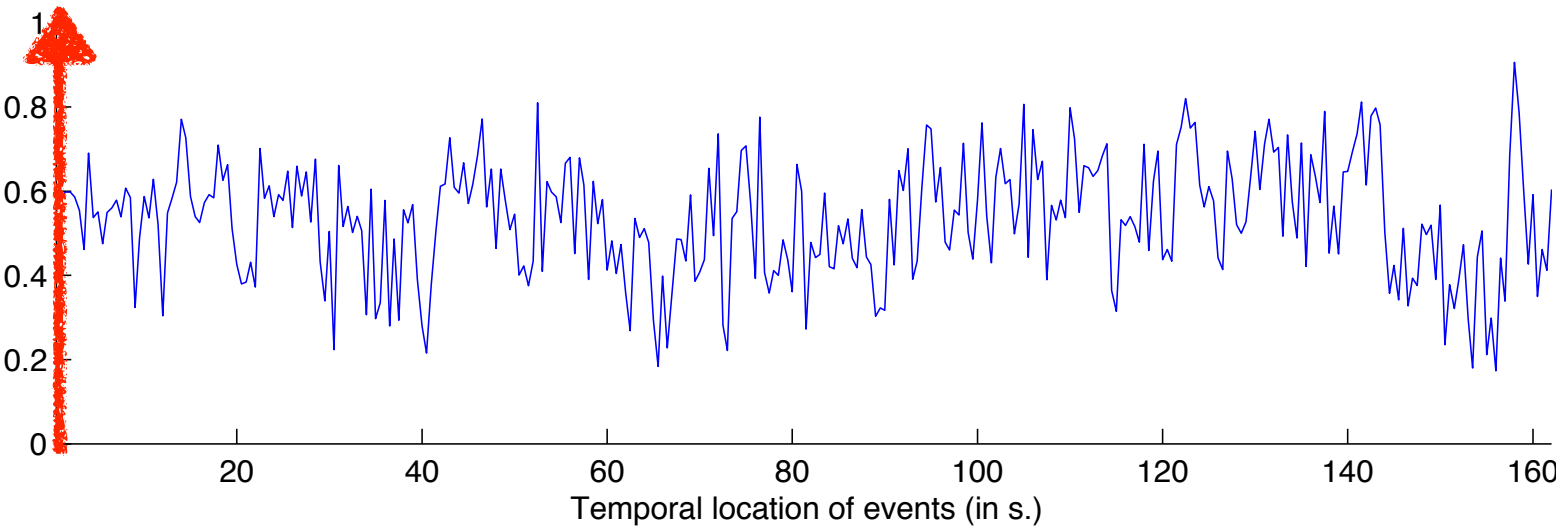
unclear



Tchaïkovski, *Swan Lake*, Act one

clear

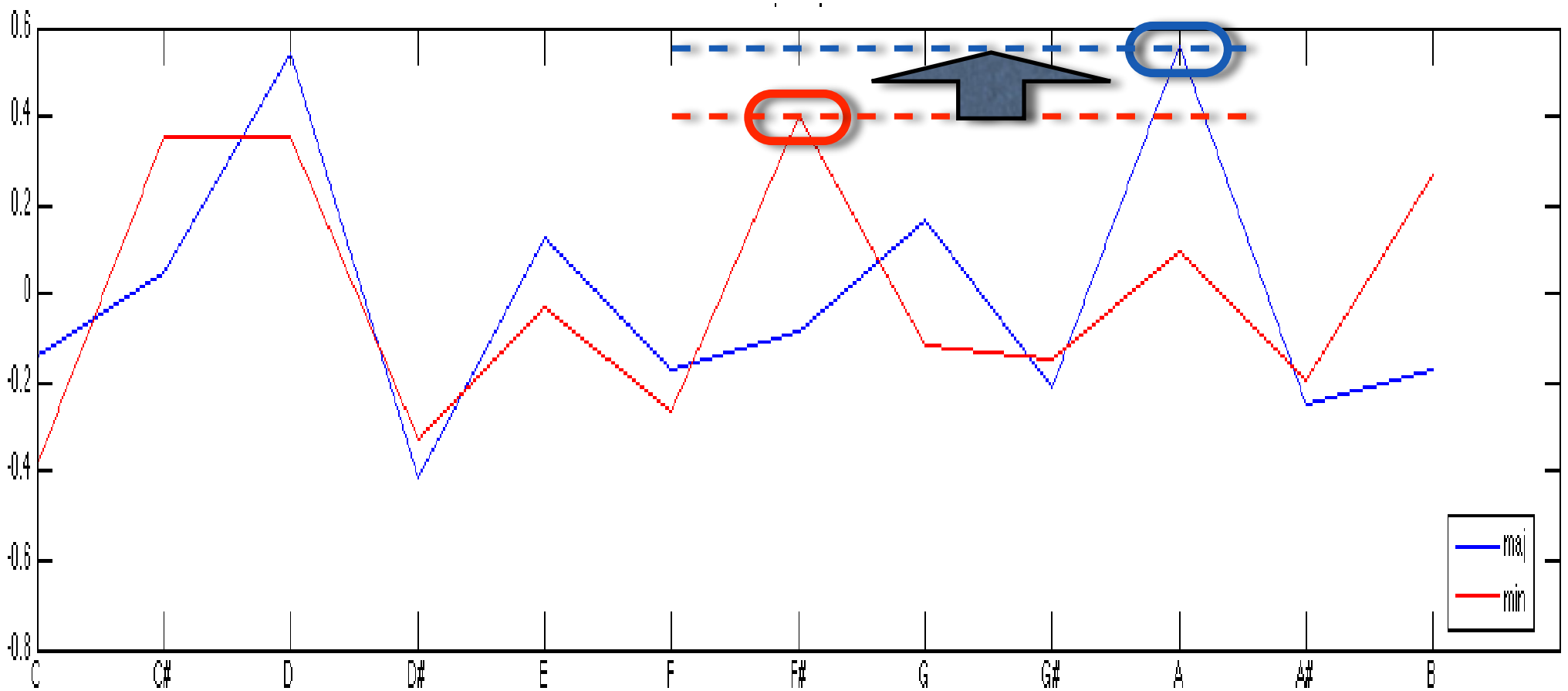
unclear



Prokofiev, Violon concerto No. in D major, *Scherzo: Vivacissimo*

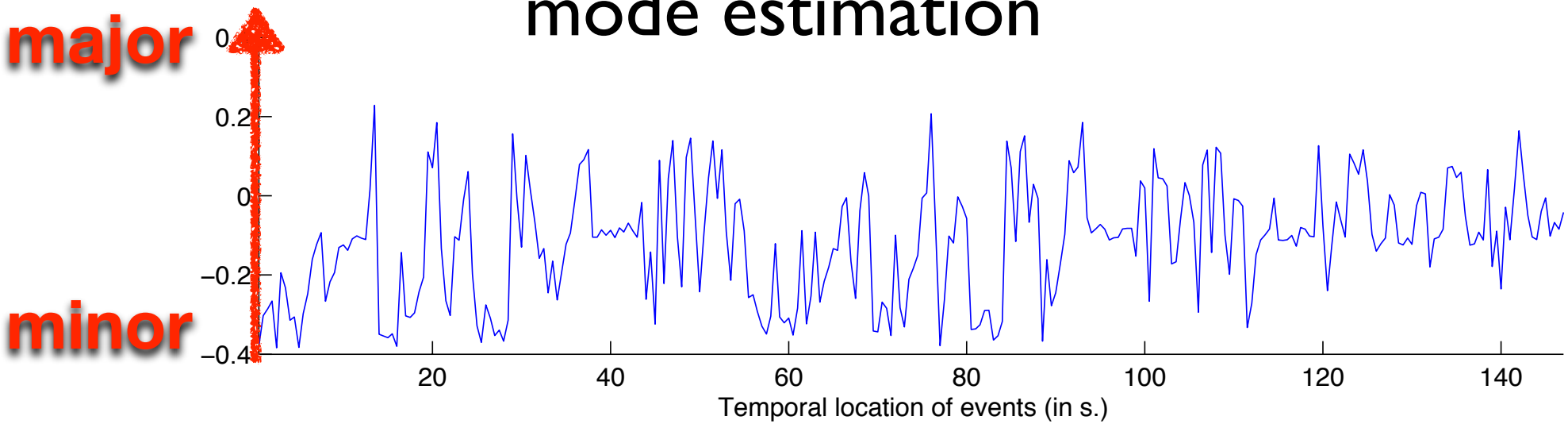
*mir*mode

mode estimation

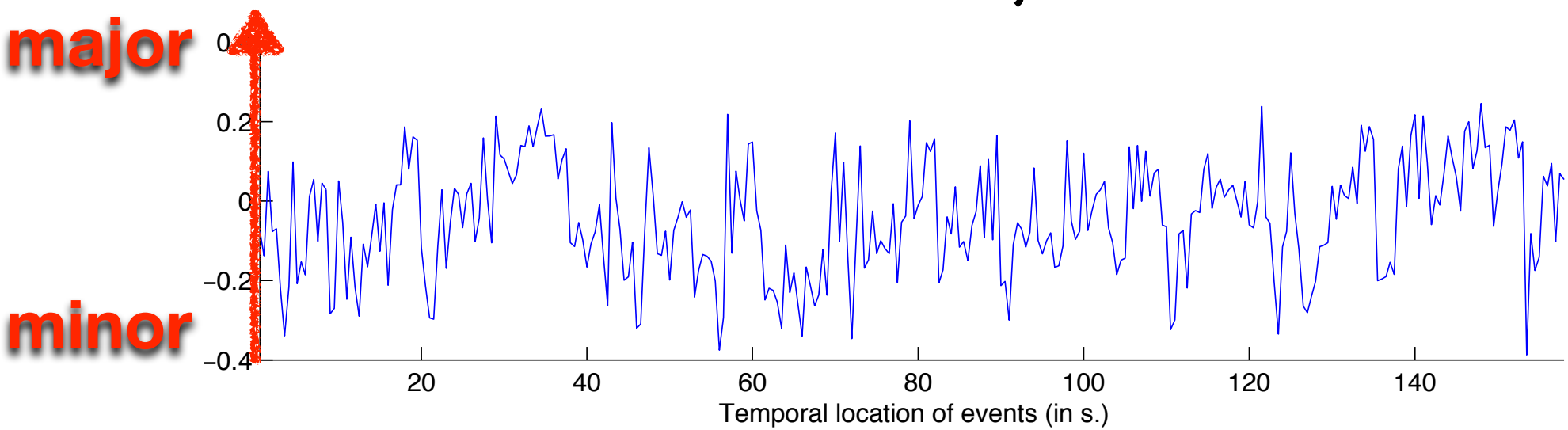


*mir*mode

mode estimation



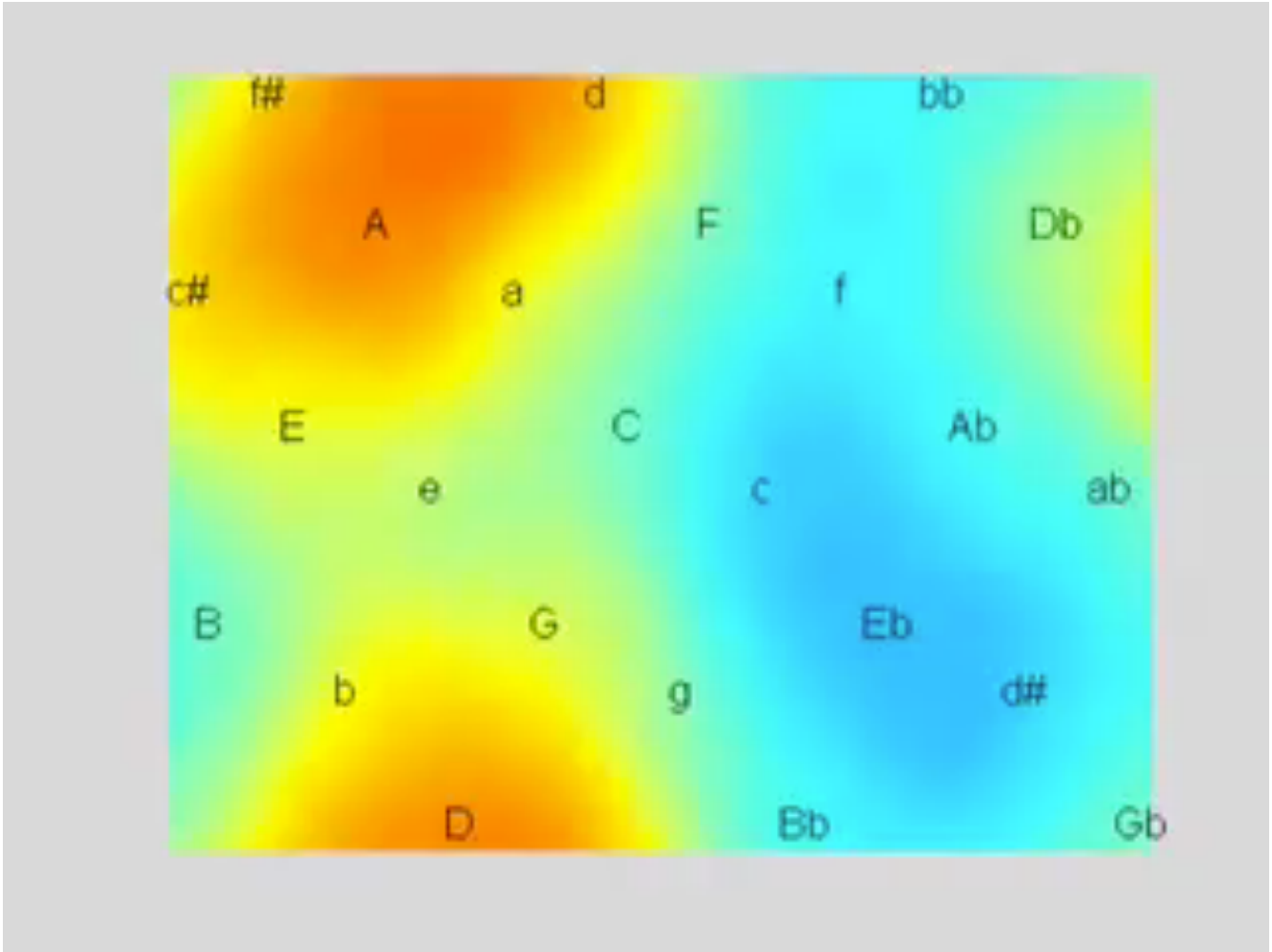
Schubert, Piano Trio No.2 in E-flat major, *Andante con moto*



Mozart, Piano Concerto No.23 in A major, *Adagio*

*mir*keysom

self-organizing map projection of chromagram

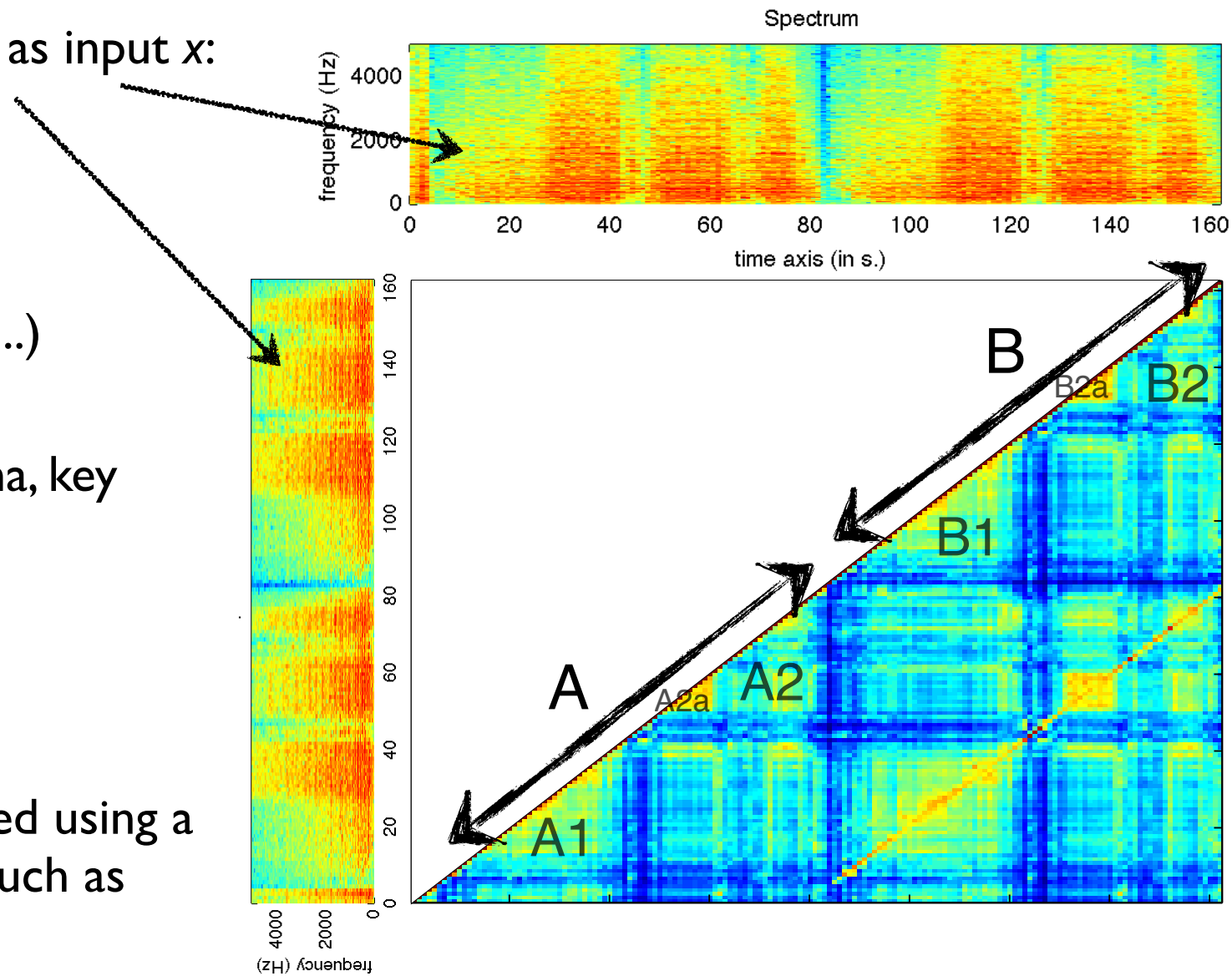


Toiviainen & Krumhansl, "Measuring and modeling real-time responses to music: The dynamics of tonality induction", *Perception* 32-6, pp. 741–766, 2003.

*mir*simatrix

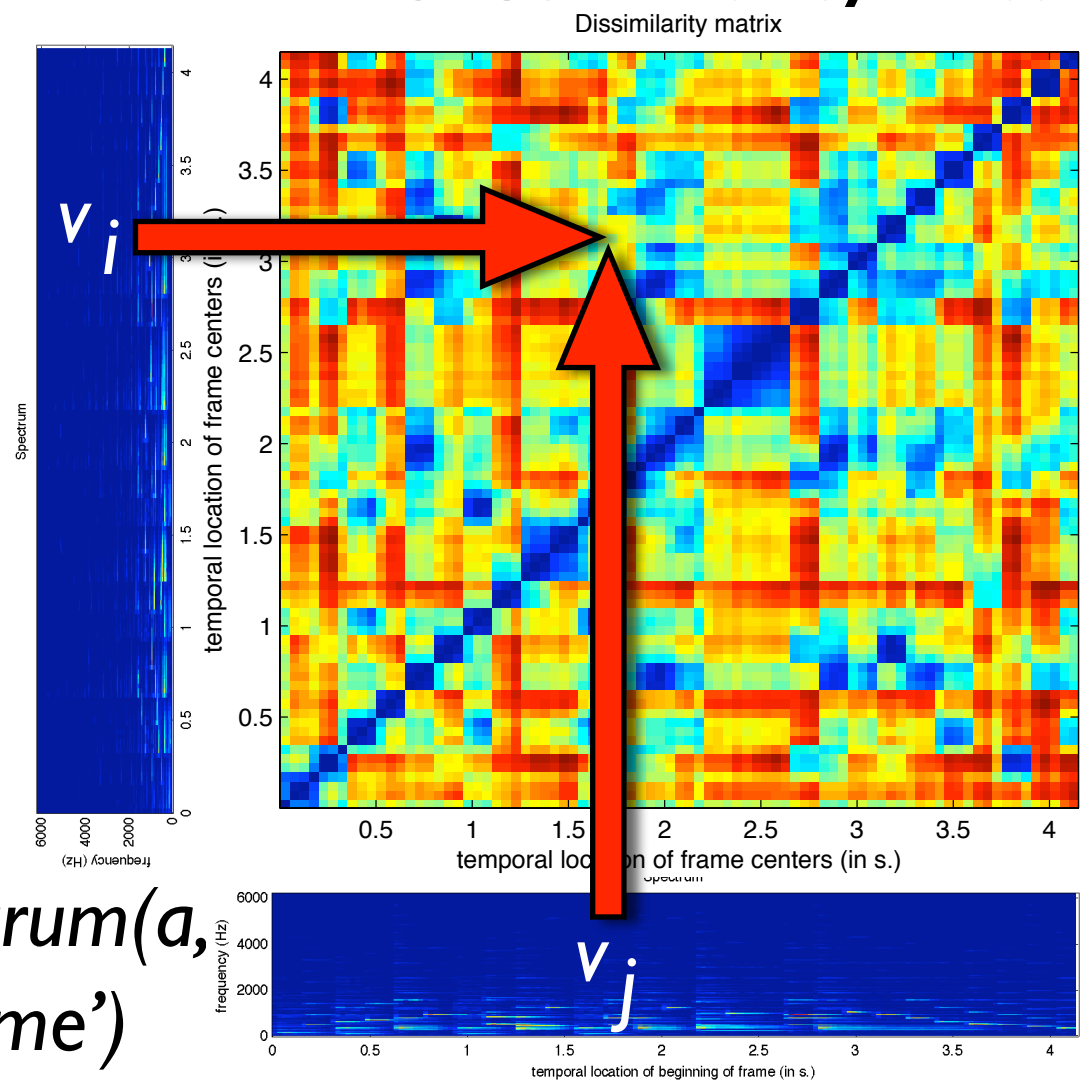
similarity matrix

- Any musical feature as input *x*:
  - Spectrum
  - Timbre(MFCC, ...)
  - Tonality (Chroma, key strength, ...)
  - Rhythm, etc.
- Frames are compared using a distance measure (such as cosine distance)



*mir*simatrix

dissimilarity matrix



mirsimatrix(a, ***'Dissimilarity'****)*

mirsimatrix(..., ***'Distance'****,* *'cosine')*

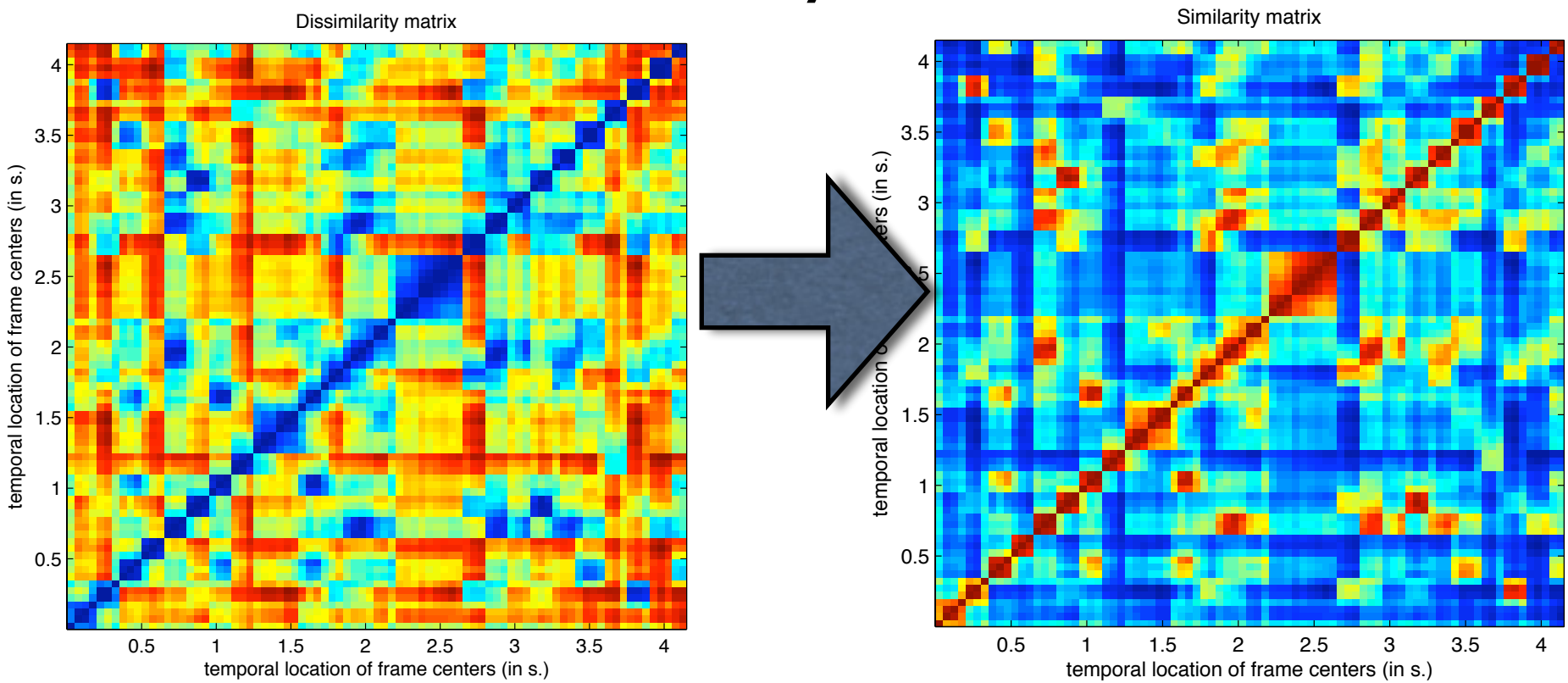
$$d_{cos}(v_i, v_j) = \frac{< v_i, v_j >}{|v_i||v_j|}$$

mirspectrum(a, 'Frame')

Foote, Cooper. "Media Segmentation using Self-Similarity Decomposition", *SPIE Storage and Retrieval for Multimedia Databases*, 5021, 167-75.

mirsimatrix

similarity matrix



mirsimatrix(a, 'Similarity', 'exponential')

$$d_{exp}(v_i, v_j) = \exp\left(-d_{cos}(v_i, v_j)\right)$$

Foote, Cooper. "Media Segmentation using Self-Similarity Decomposition", *SPIE Storage and Retrieval for Multimedia Databases*, 5021, 167-75.

mirsimatrix

similarity matrix

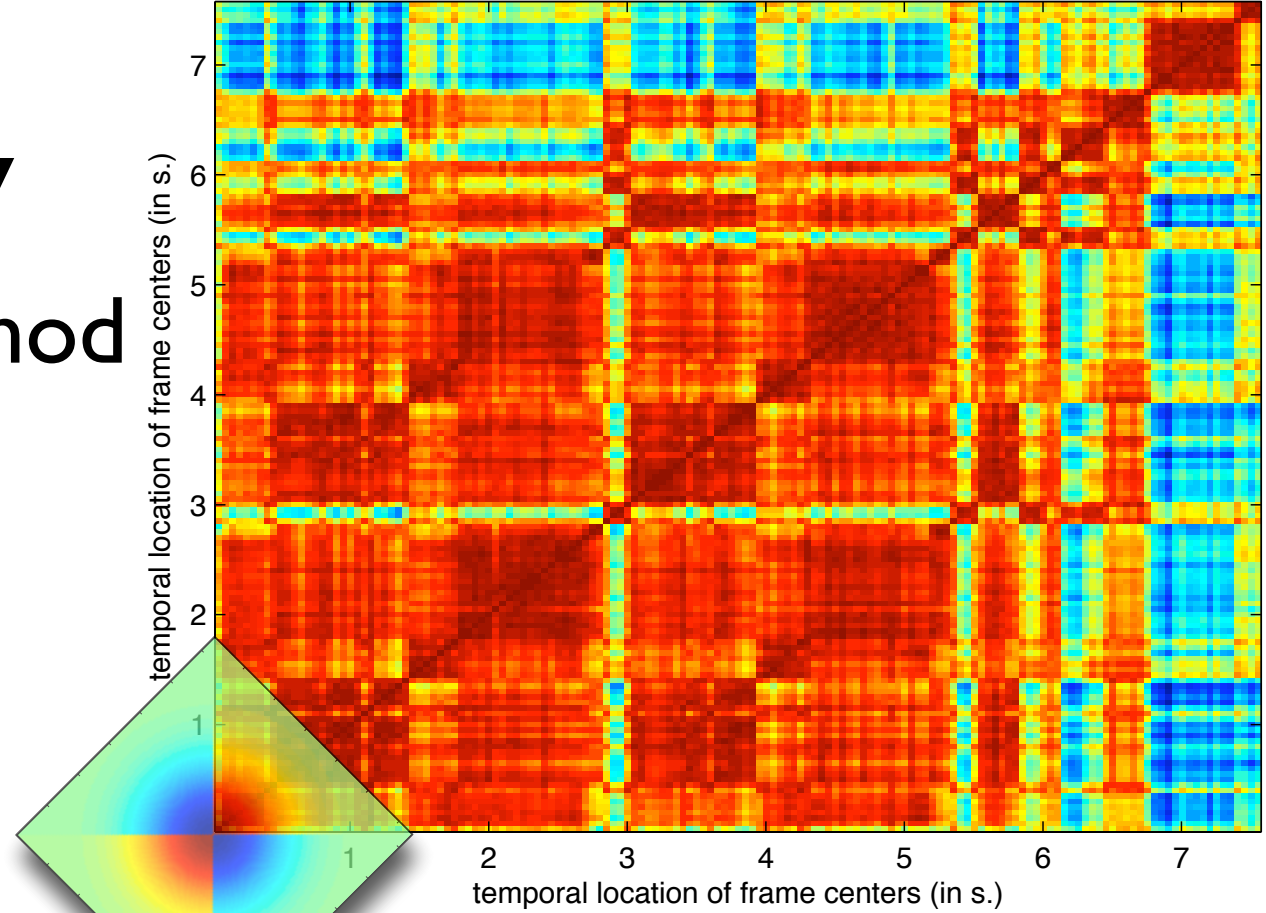
- For instance:
  - *s = mirspectrum(…, 'Frame', …)*
  - *mirsimatrix(s)*

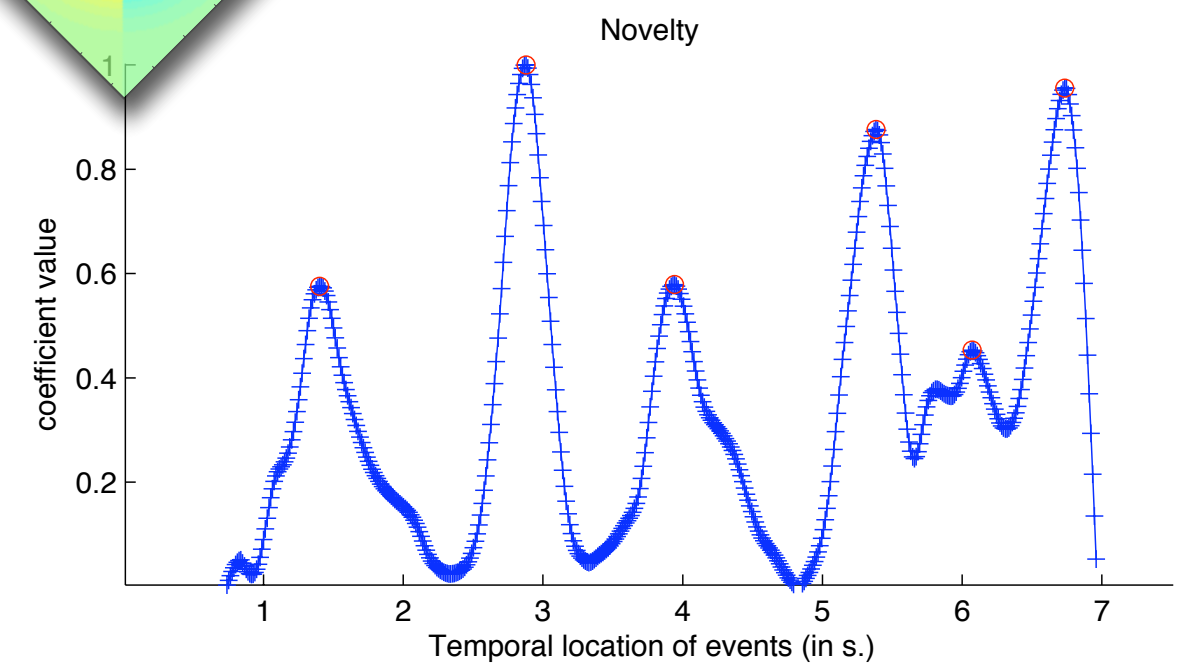
- Observe the structure of this excerpt along different musical dimensions:
  - '*george.wav*'

*mir*novelty

novelty, kernel method

- Convolution with checkerboard kernel along the diagonal of *mirsimatrix*
- Peaks indicate transitions between phases





*mir**novelty***

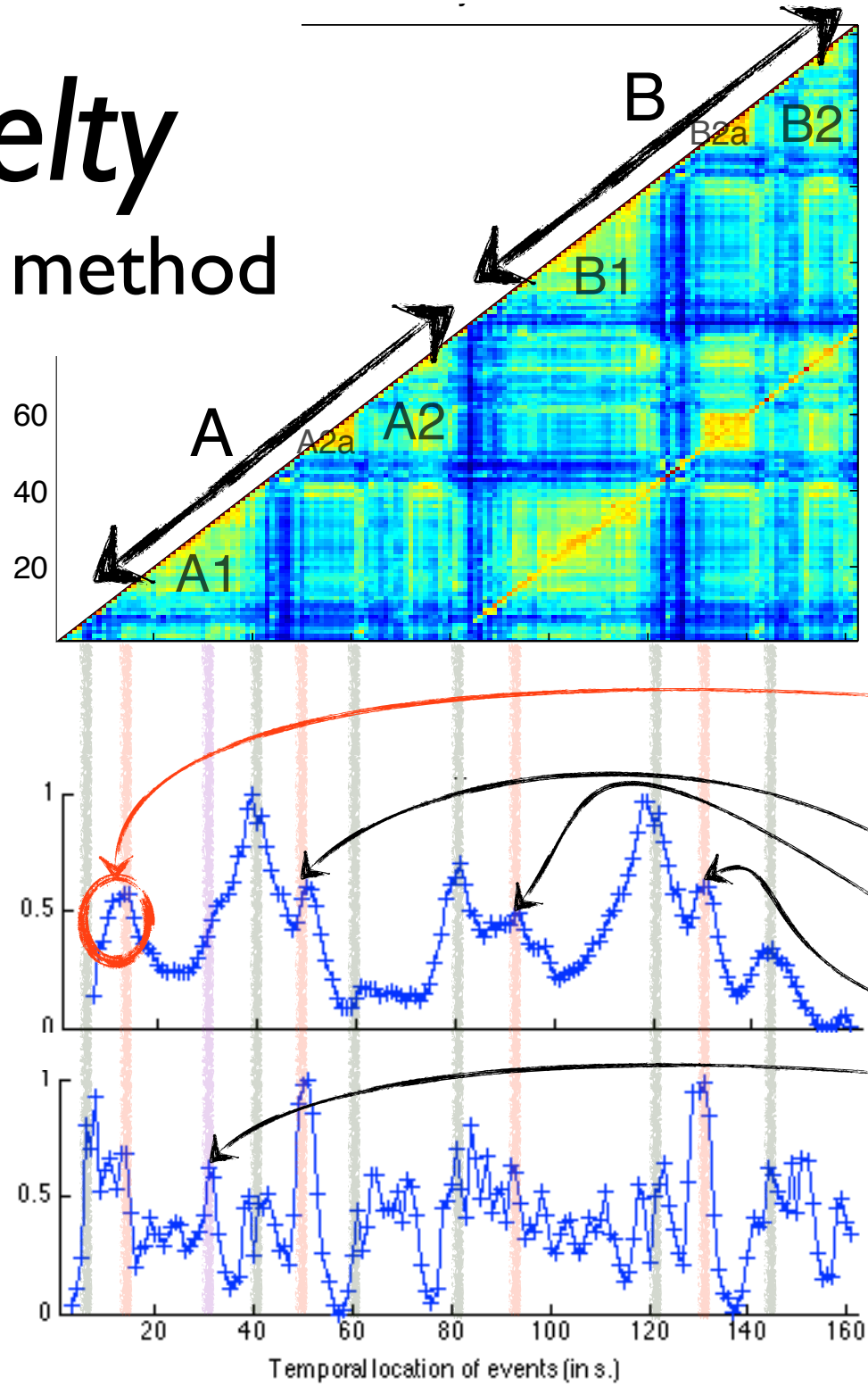
novelty, kernel method

- *mirnovelty(…, '**Kernel**', N)*
  - where N is the kernel size

- Observe the structure of this excerpt using different kernel sizes:
  - '*george.wav*'

*mir*novelty

novelty, kernel method



Kernel size:

- 64 frames
- 16 frames

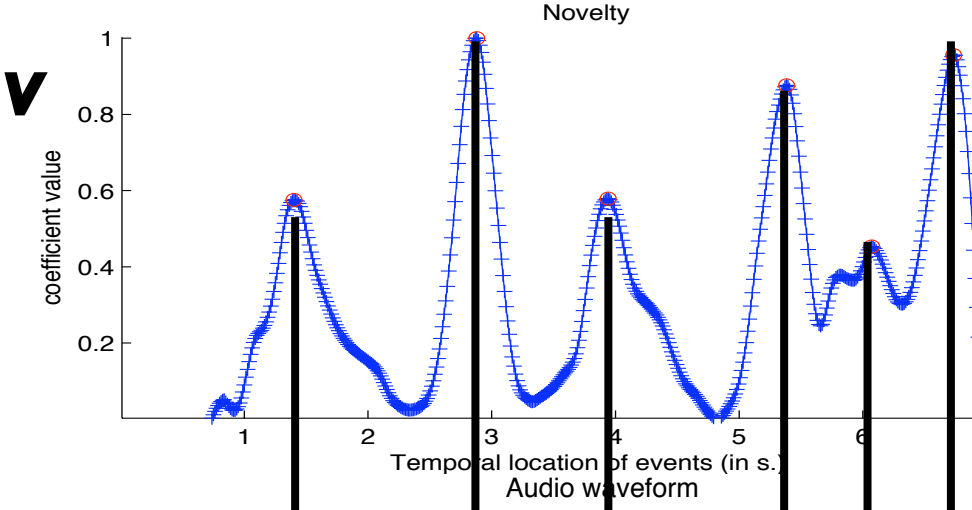
- end of intro **& start** of A1
- **start** of A2a
- **start** of B1
- **start** of B2a
- transition A1a/A1b

*mir*segment

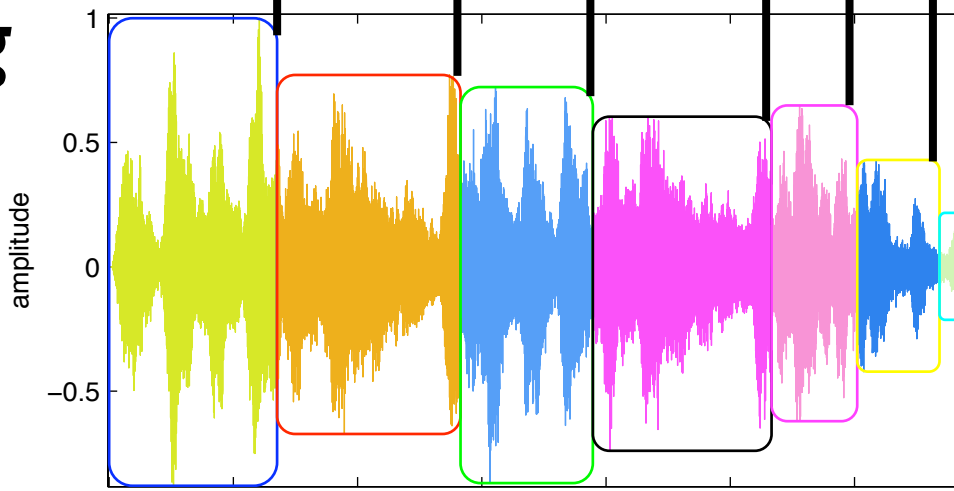
segmentation

- ***nv*** = *mirnovelty(sm)*
- ***p*** = *mirpeaks(nv)*
- ***sg*** = *mir**segment**('mysong', **p**)*
- *mirplay(**sg**)*
- ***s*** = *mirmfcc(**sg**)*
- ***sm*** = *mirsimatrix(**s**)*

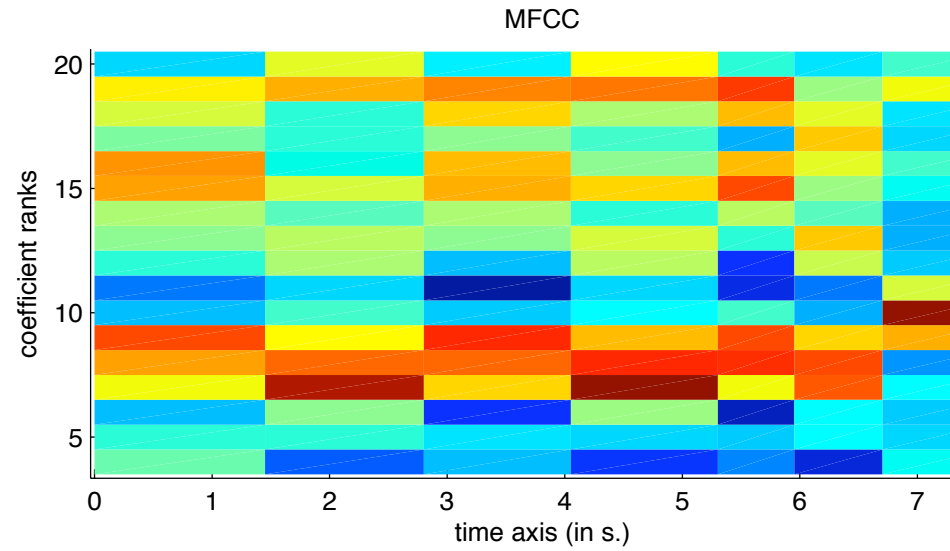
nv



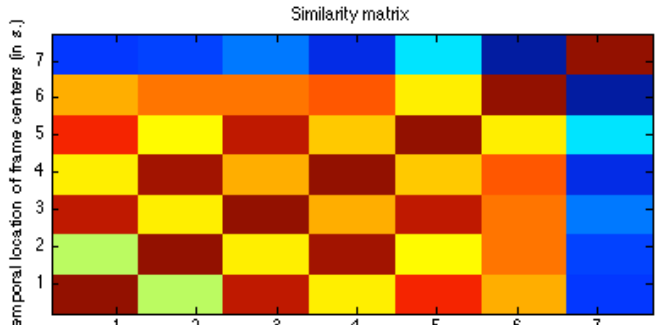
sg



s

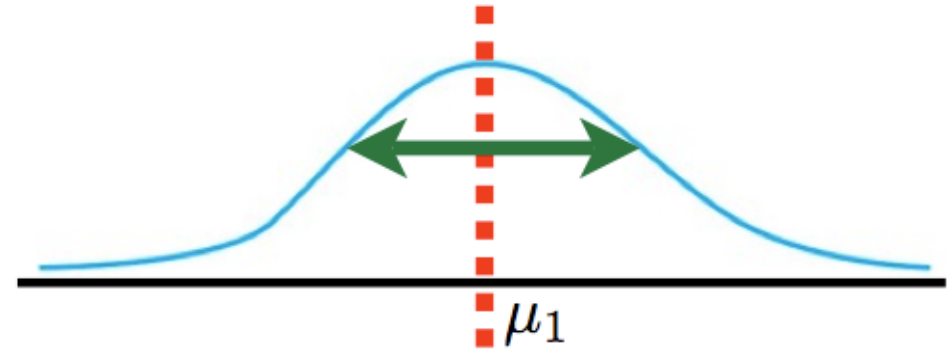


sm

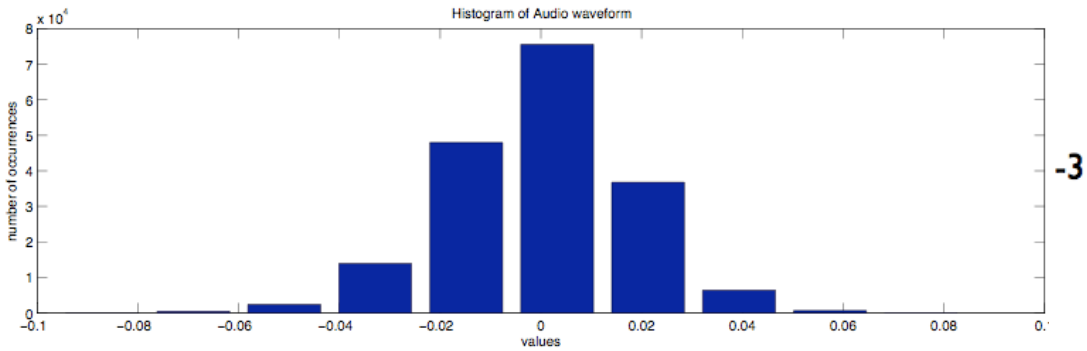


Statistics

- *mir**stat***
  - mean
  - standard deviation
  - slope
  - periodicity
- *mir**histo***
  - distribution histograms



- moments
  - *mir**centroid***
  - *mir**spread***
  - *mir**skewness***
  - *mir**kurtosis***



mirfeatures

batch of features

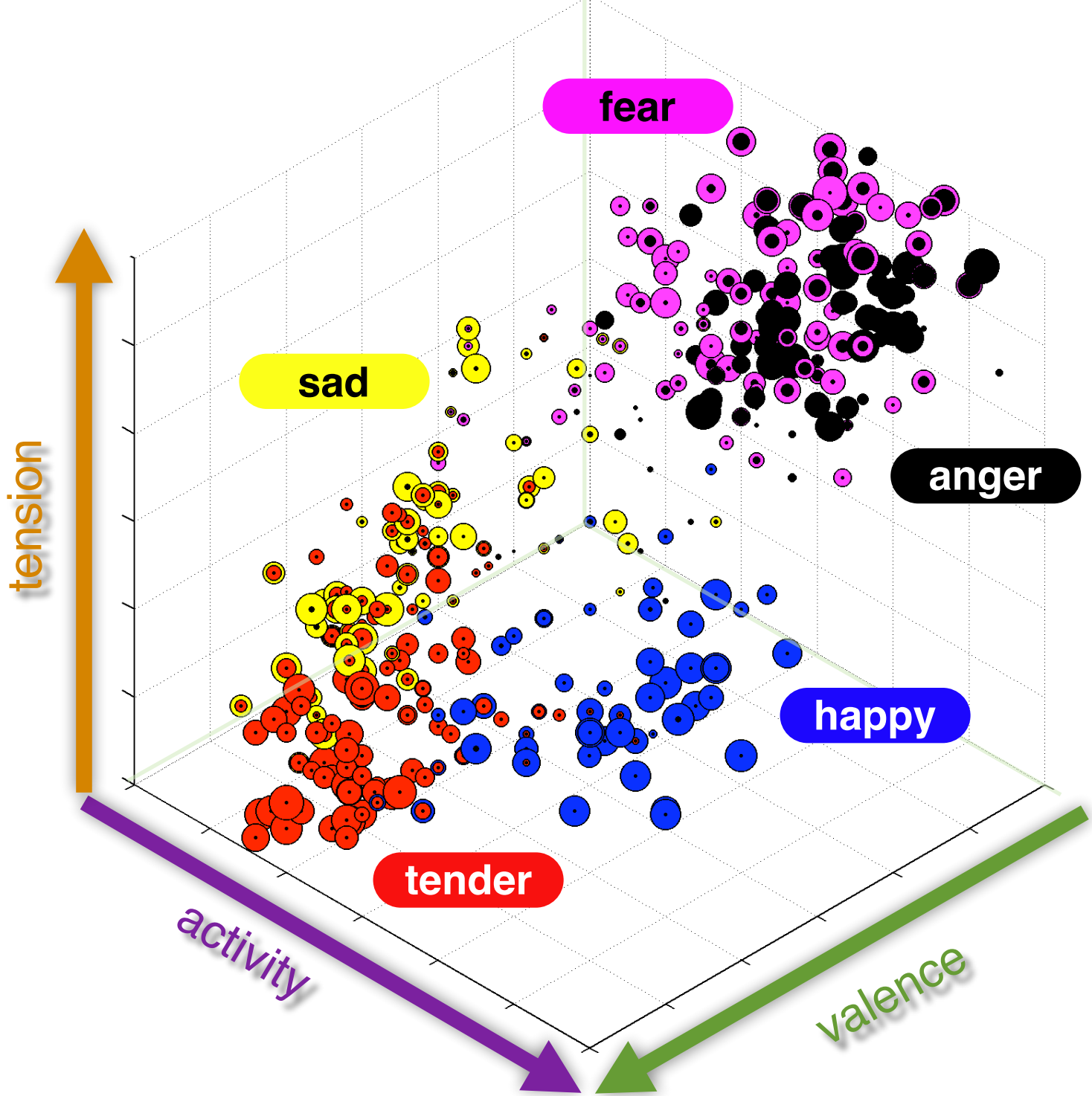
- *mirzerocross*
- *mircentroid*
- *mirbrightness*
- *mirspread*
- *mirskewness*
- *mirkurtosis*
- *mirrolloff*
- *mirentropy*
- *mirflatness*
- *mirroughness*
- *mirregularity*
- *mirinharmonicity*
- *mirmfcc*
- *mirfluctuation*
- *mirattacktime*
- *mirattackslope*
- *mirlowenergy*
- *mirflux*
- *mirpitch*
- *mirchromagram*
- *mirkeystrength*
- *mirkey*
- *mirmode*
- *mirhcdf*
- *mirtempo*
- *mirpulseclarity*

*mirfeatures('Folder', '**Stat**')*

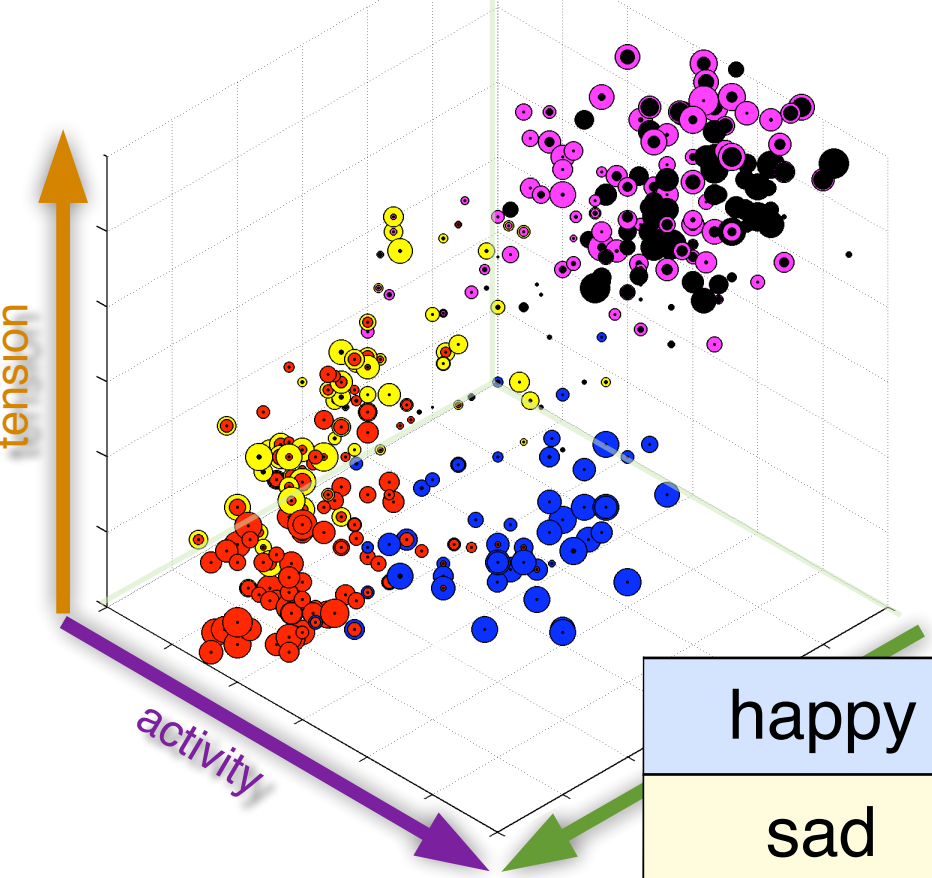
mirexport

exportation of statistical data to files

- *mirexport(filename,* ***...****)* adding one or several data from *MIRtoolbox* operators.
- *mirexport('result****.txt****', ...)* saved in a text file.
- *mirexport('result****.arff****', ...)* exported to WEKA for data-mining.
- *mirexport('****Workspace****', ...)* saved in a Matlab variable.



Eerola & Vuoskoski. A comparison of the discrete and dimensional models of emotion in music. Psychology of Music.



	3D				2D		
	R^2	β			R^2	β	
		v	*a*	*t*		*v*	*a*
happy	.89	.93	.79	-.35	.89	.85	.49
sad	.63	-.20	-.84	-.22	.63	-.05	-.69
tender	.77	.33	-.45	-.58	.77	.50	-.51
fear	.87	-.83	.07	.63	.87	-.90	.24
anger	.64	-.52	.32	.35	.64	-.55	.35
mean	.76				.76		

Eerola & Vuoskoski. A comparison of the discrete and dimensional models of emotion in music. Psychology of Music.

miremotion

R^2	h	s	t	f	a
MLR	.55	.56	.42	.51	.54
PCA	.34	.40	.23	.13	.27
PLS	.59	.64	.52	.60	.57
MLR	.62	.58	.44	.60	.63
PCA	.38	.49	.47	.45	.62
PLS	.65	.61	.55	.64	.67

Box-Cox

happy	
Feature	**β**
fluctuation	-.40
sp. centroid	.13
sp. spread	-.19
chrom. peaks	-.05
majorness	.03

sad	
Feature	**β**
roughness	.12
register	-.08
register var	.09
majorness	.02
harm. change	-.03

tender	
Feature	**β**
RMS var	-.42
sp. centroid	.14
key clarity	.11
harm. change	-.10
tonal novelty	-.01

fear	
Feature	**β**
RMS var	-.79
fluctuation	-.21
key clarity	-.09
harm. change	.08
tonal novelty	-.02

anger	
Feature	**β**
RMS var	.44
pulse clarity	-.13
sp. centroid	-.13
key clarity	-.04
tonal novelty	.02

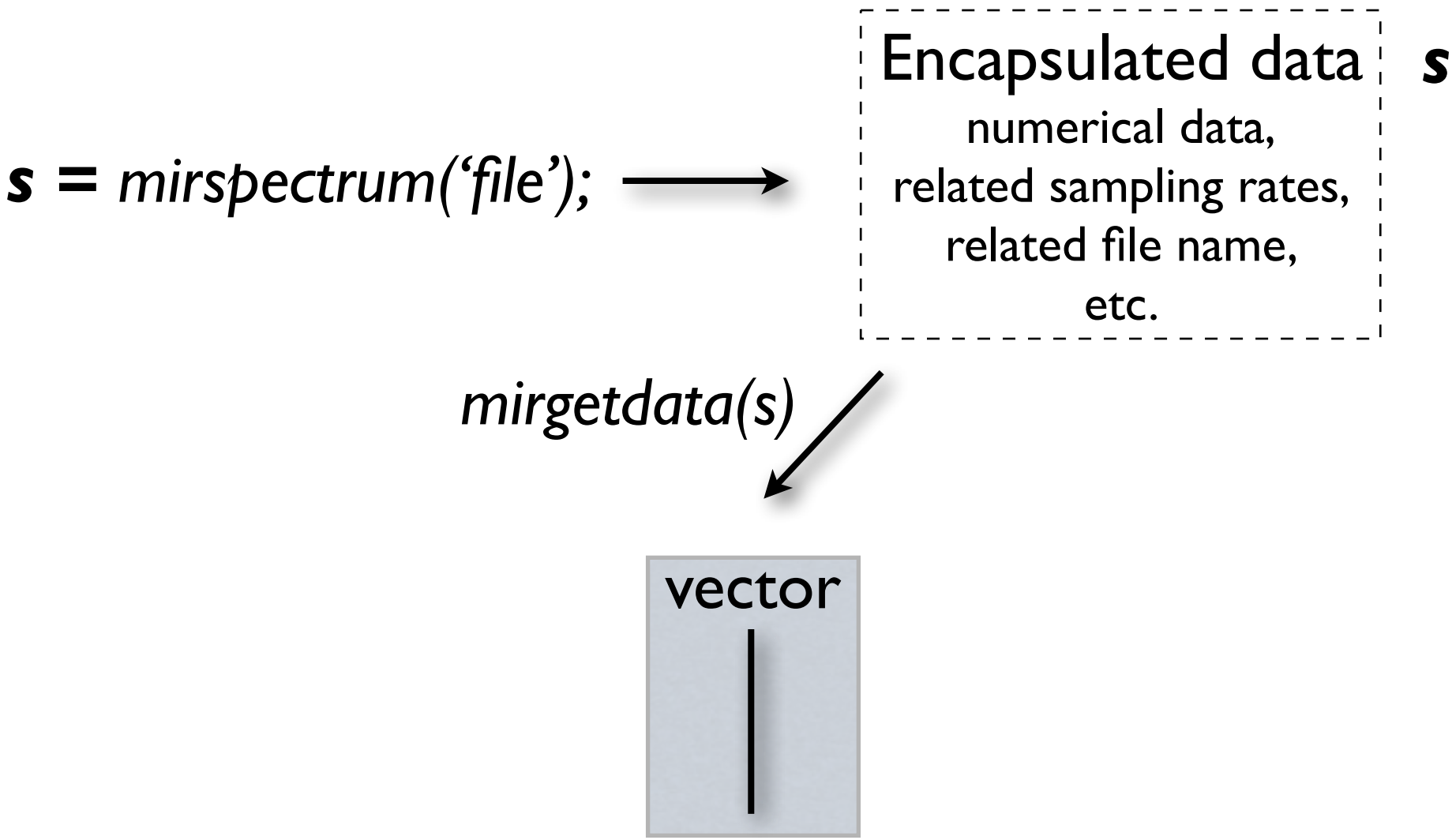
T. Eerola, O. Lartillot, P. Toiviainen, "Prediction of Multidimensional Emotional Ratings in Music From Audio Using Multivariate Regression Models", *ISMIR*, Kobe, 2009.

Part 2

- Rhythm, metrical structure
- Tonal analysis
- Segmentation, structure
- Statistics
- Music & emotion
- **Advanced use**

*mir*getdata

return data in Matlab format



*mir*getdata

return data in Matlab format

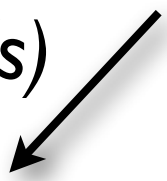
*s = mirspectrum('file', '**Frame**');*

Encapsulated data

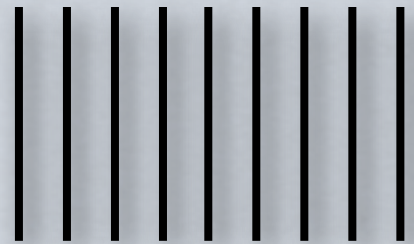
grouped numerical data,
related sampling rates,
related file name,
etc.

s

mirgetdata(s)



matrix



*mir*getdata

return data in Matlab format

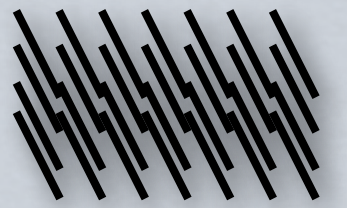
*f = mirfilterbank('file', '**Frame**');*

Encapsulated data *f*

grouped numerical data,
related sampling rates,
related file name,
etc.

mirgetdata(f)

3D-matrix



*mir*getdata

return data in Matlab format

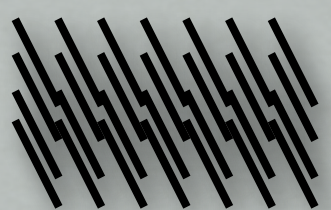
sg = mirsegment('file')
s = mirspectrum(sg, 'Frame');

Encapsulated data **s**
grouped numerical data,
related sampling rates,
related file name,
etc.

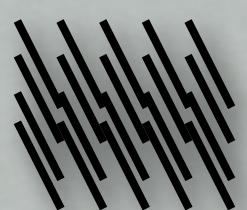
mirgetdata(s)

cell array

3D-matrix



3D-matrix



...

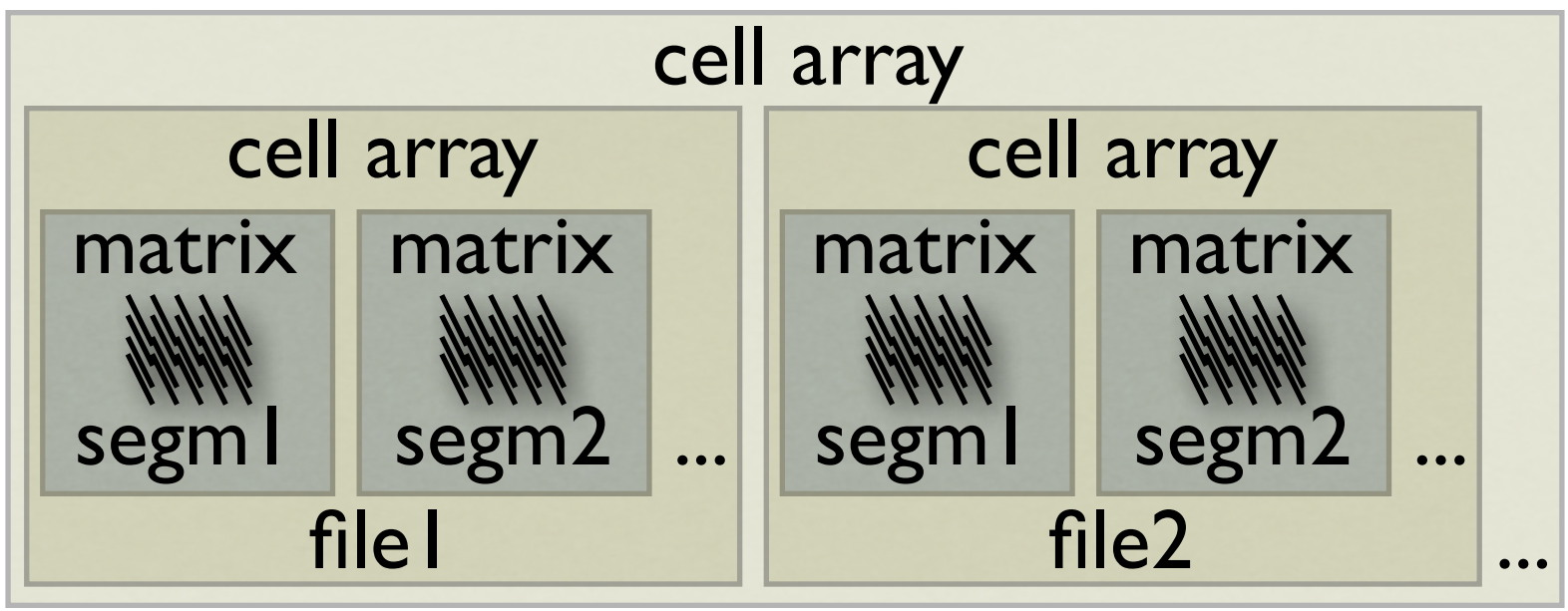
*mir*getdata

return data in Matlab format

*sg = mirsegment('**Folder**')*
s = mirspectrum(sg,'Frame');

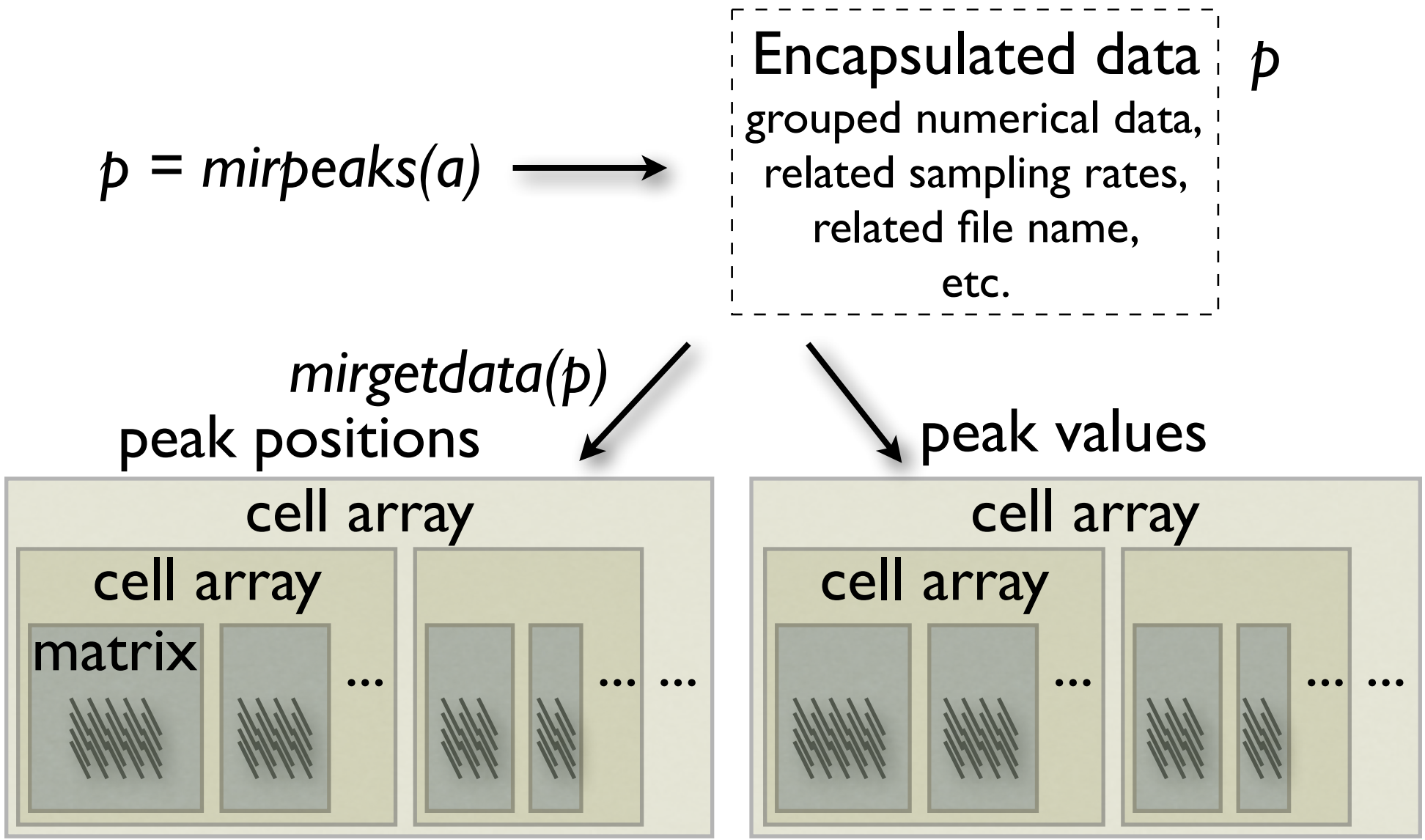
Encapsulated data `s`
grouped numerical data,
related sampling rates,
related file name,
etc.

mirgetdata(s)



*mir*getdata

return data in Matlab format



get

returns fields of encapsulated data

a = miraudio('ragtime'); ⟶ Encapsulated data

- *get(a,* '***xName***')
- *get(a,* '***xData***')
- *get(a,* '***yName***')
- *get(a,* '***yData***')
- *get(a,* '***yUnit***')
- *get(a,* '***FramePos***')
- *get(a,* '***Sampling***')
- *get(a,* '***NBits***')
- *get(a,* '***Title***')
- *get(a,* '***FileName***')
- *get(a,* '***Label***')
- *get(a,* '***Channels***')
- *get(a,* '***xPeakSample***')
- *get(a,* '***xPeakUnit***')
- *get(a,* '***xPeakInterpol***')
- *get(a,* '***yPeak***')
- *get(a,* '***yPeakInterpol***')

get

returns fields of encapsulated data

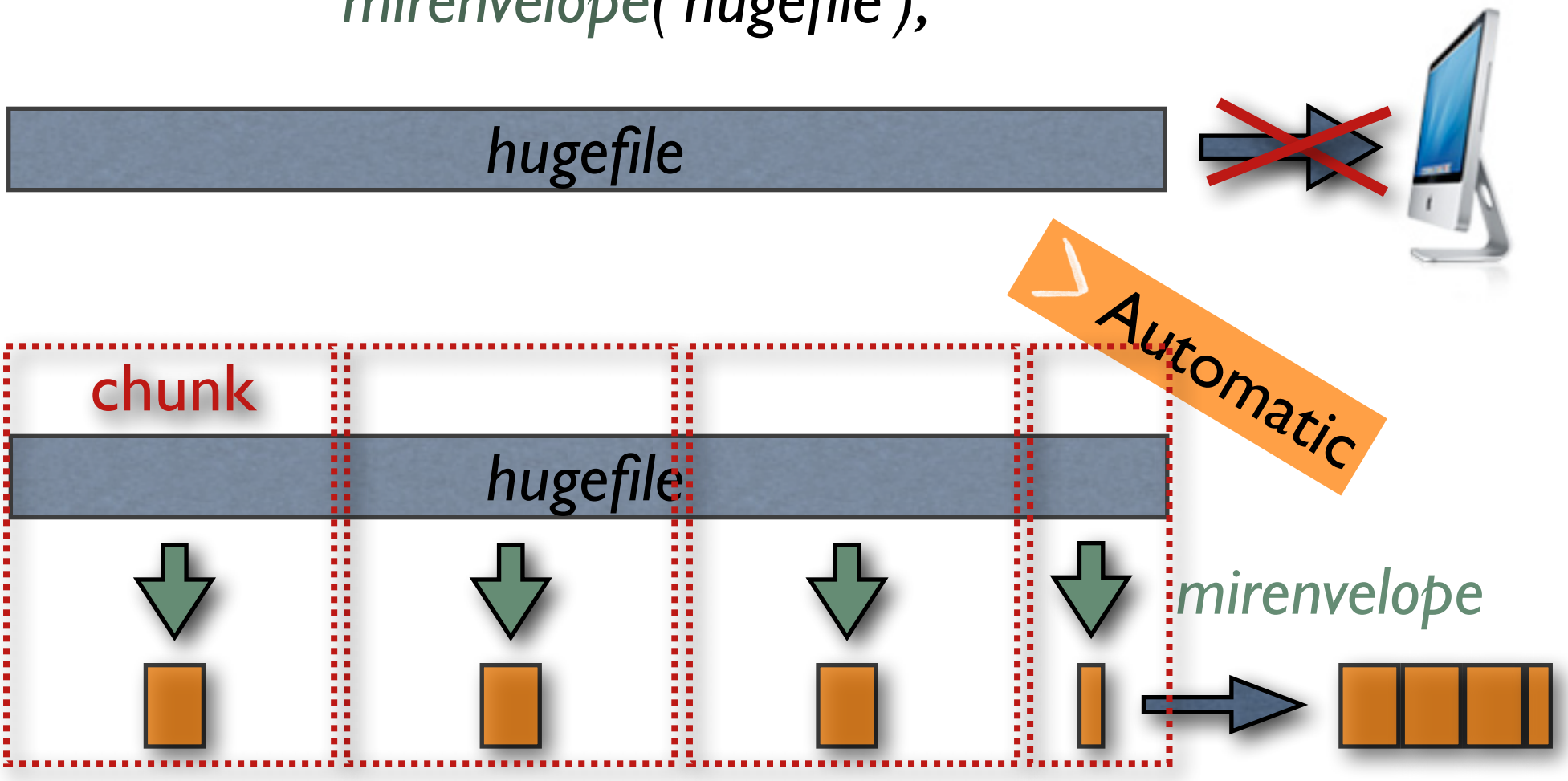
s = mirspectrum('ragtime'); ⟶ Encapsulated data

- *get(s,'**Frequency**') = get(s,'xData')*
- *get(s,'**Magnitude**') = get(s,'yData')*
- *get(s,'**Phase**')*
- *get(s,'**xScale**')* (= *'Freq','Mel','Bark'*)
- *get(s,'**Power**')*
- *get(s,'**dB**')*

etc.

memory management

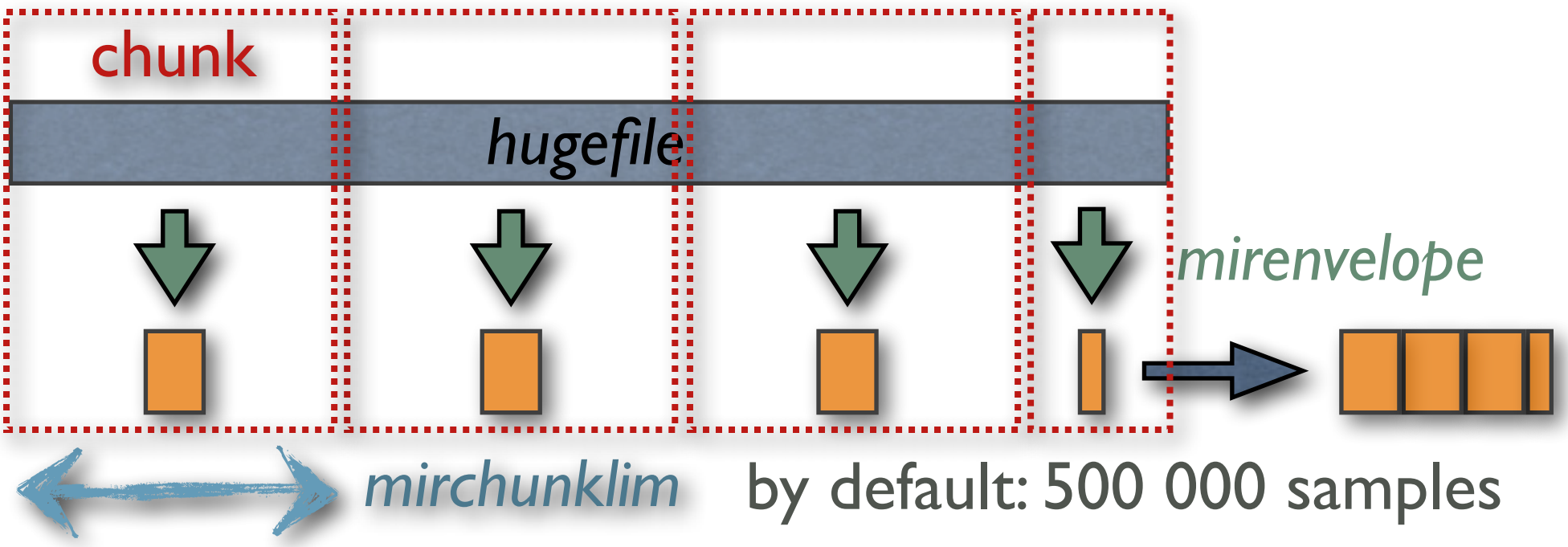
mirenvelope('hugefile');



mirchunklim

chunk size limitation

mirenvelope('hugefile');



by default: 500 000 samples

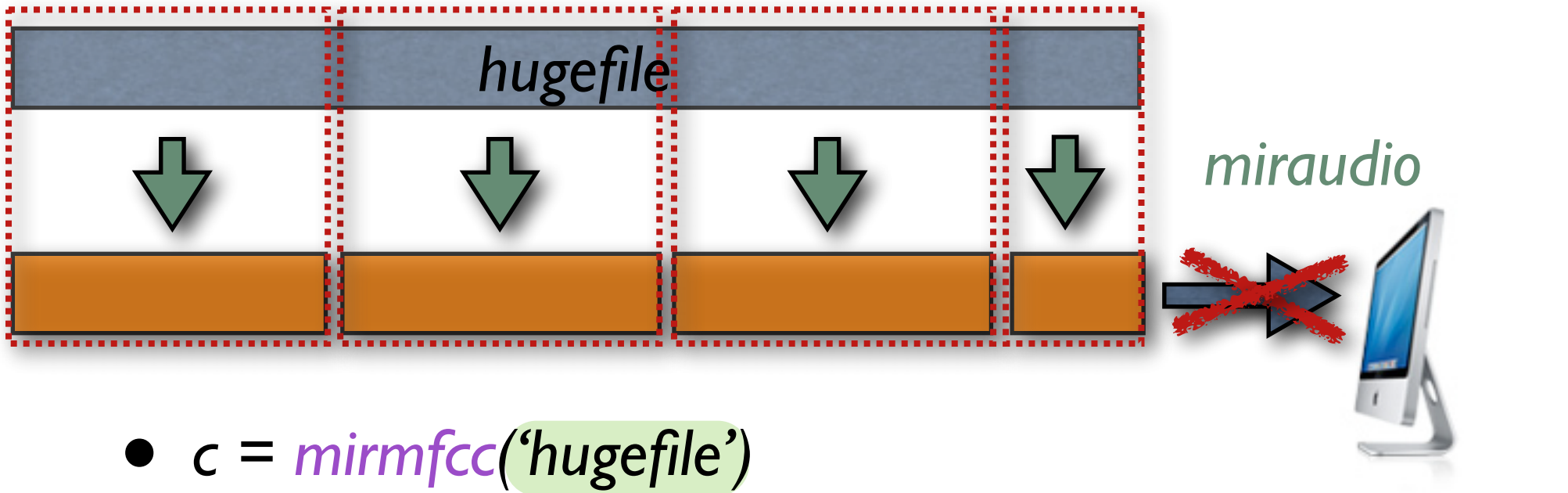
- If memory overflow problems, decrease *mirchunklim*:

mirchunklim(50000) set to 50 000 samples

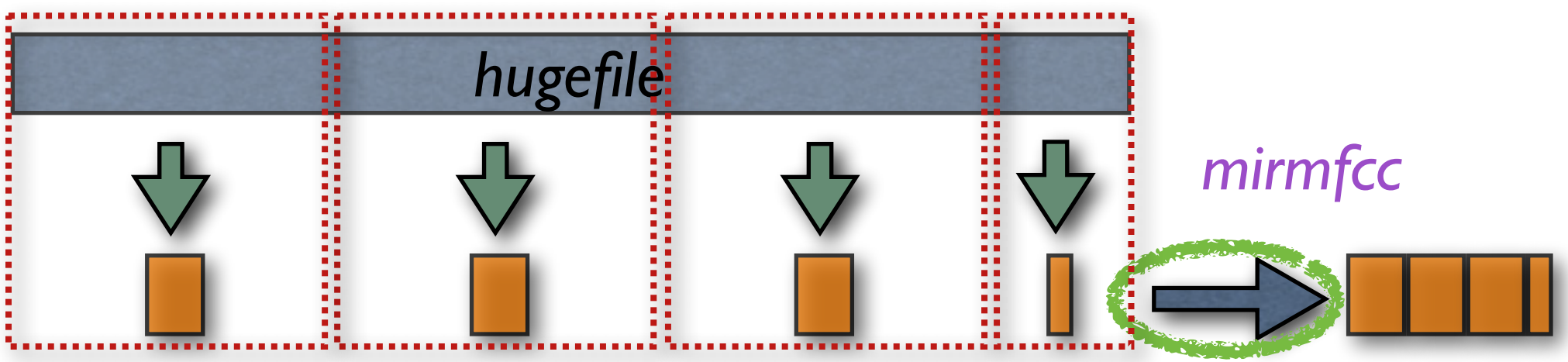
avoid useless call to *mir**audio***

- *a = miraudio('hugefile');*
- *c = mirmfcc(a)*

a
↓
c



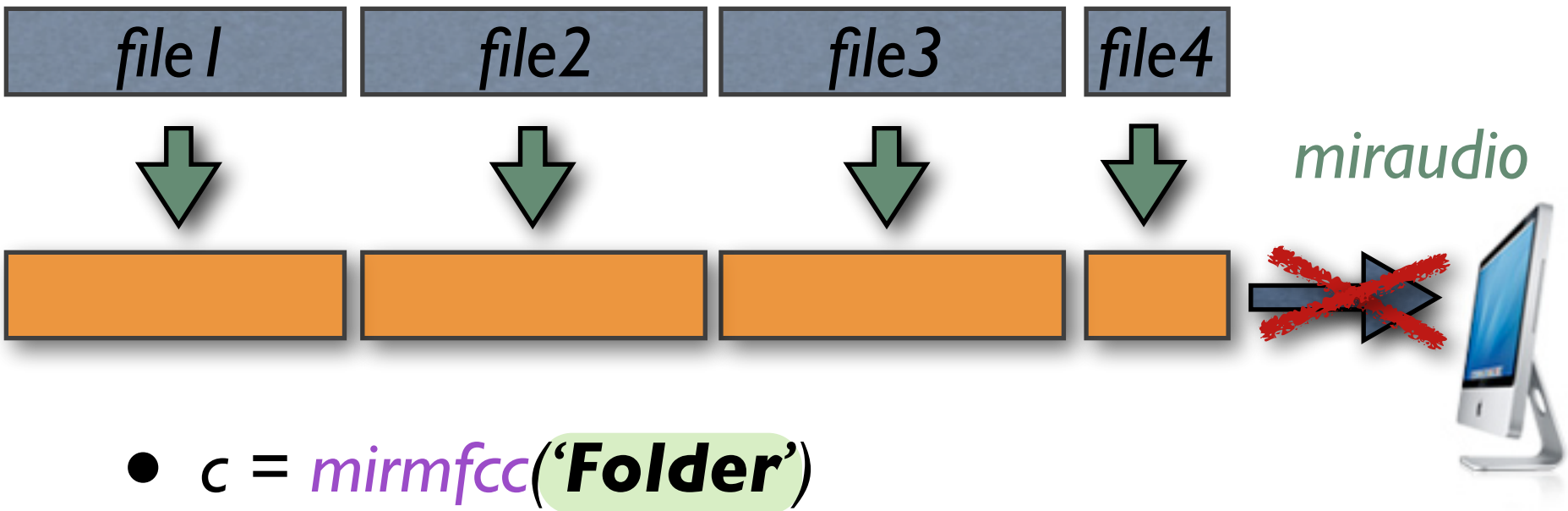
- *c = mirmfcc('hugefile')*



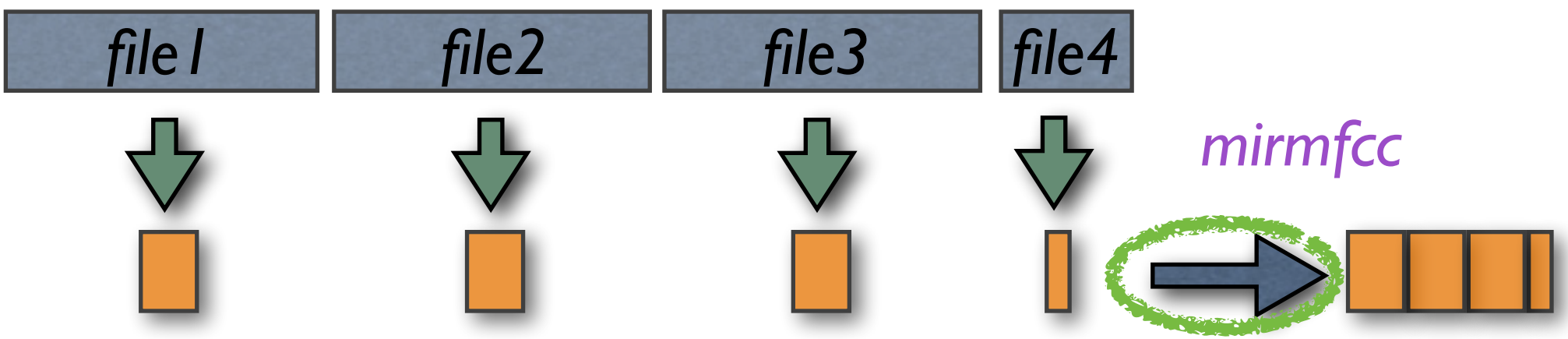
avoid useless call to *mir**audio***

- *a = miraudio(‘**Folder**’);*
- *c = mirmfcc(a)*

a
↓
c

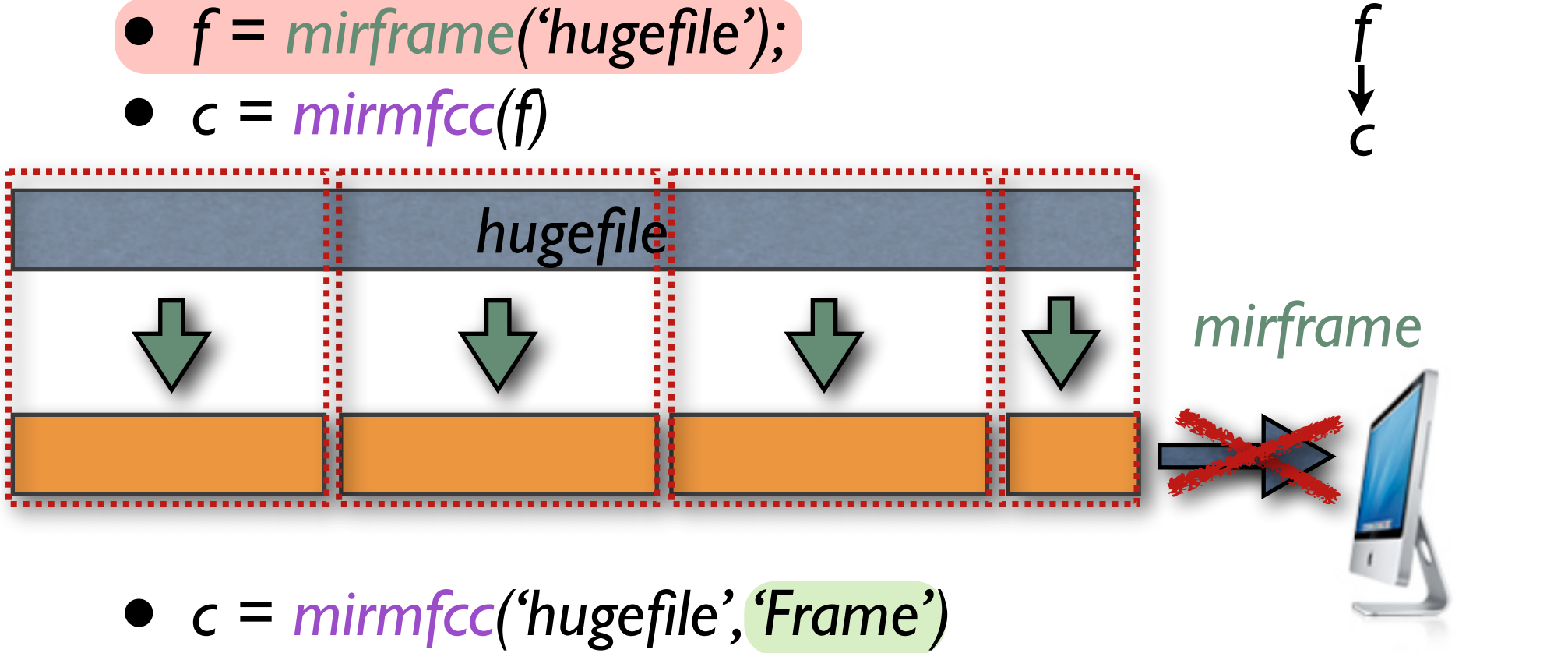


- *c = mirmfcc(‘**Folder**’)*

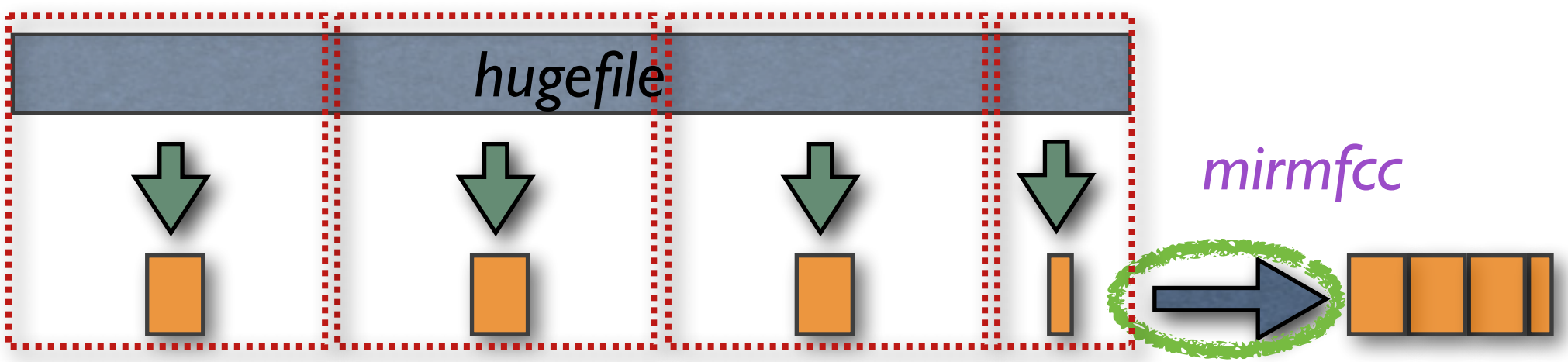


avoid useless call to *mir**frame***

- *f = mirframe('hugefile');*
- *c = mirmfcc(f)*



- *c = mirmfcc('hugefile','Frame')*

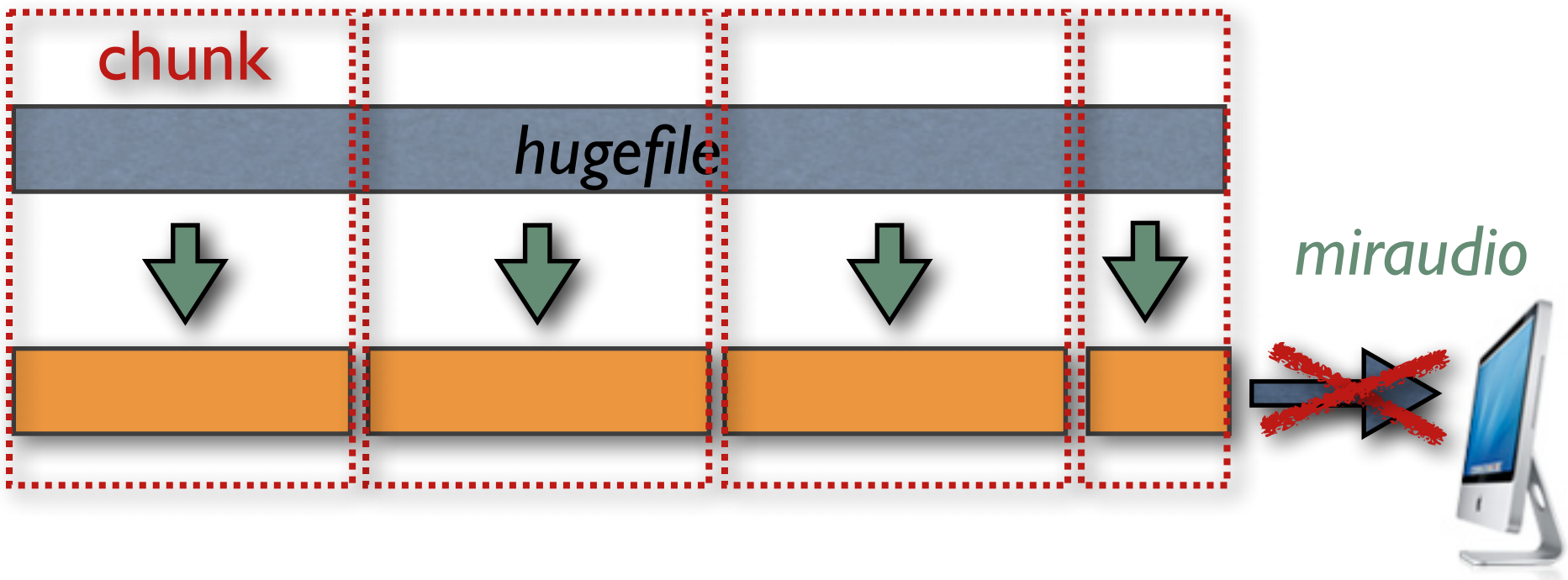


what if *miraudio* (or *mirframe*) really necessary?

?

- *a = miraudio('hugefile', 'Sampling', 11025);*
- *c = mirmfcc(a)*

a
↓
c

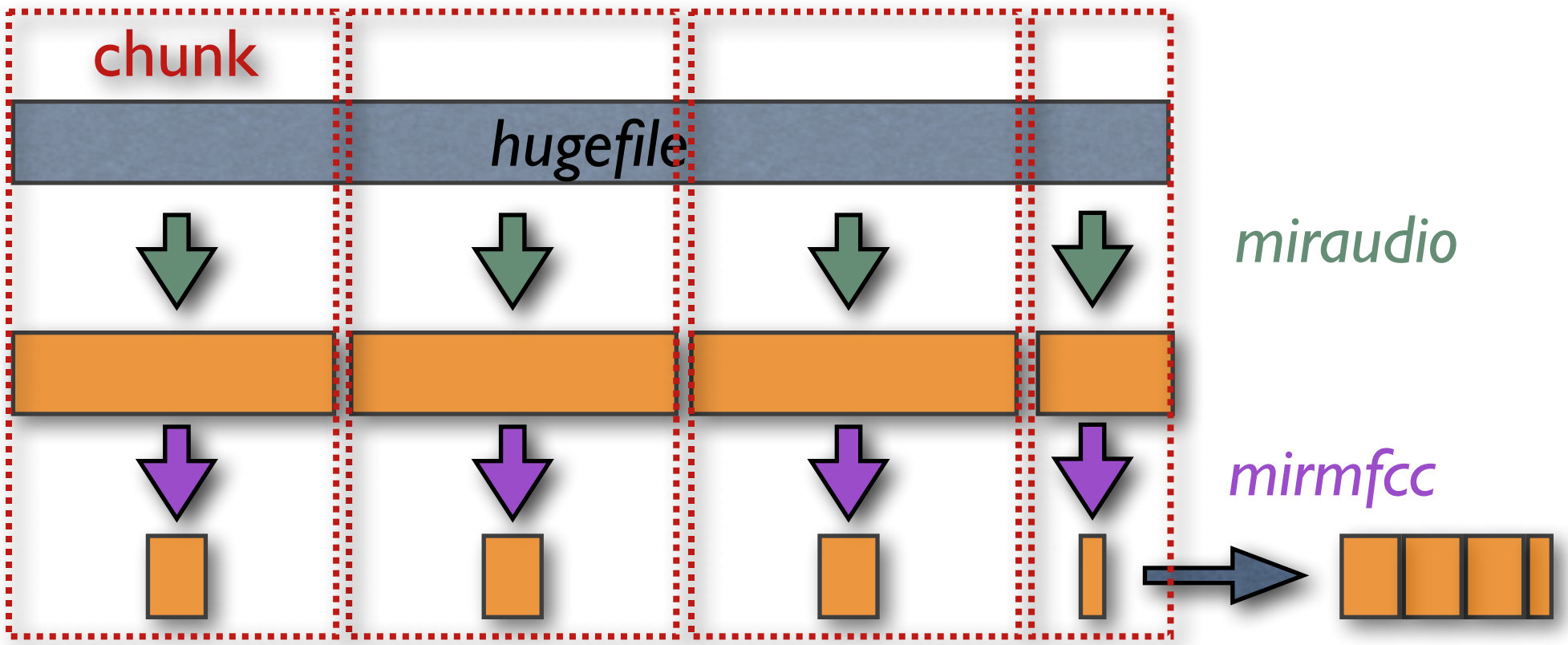


*mir**eval***

flowchart design and evaluation

- *a = miraudio('**Design**', 'Sampling', 11025);*
- *c = mirmfcc(a);*
- ***mireval**(c, '**hugefile**')*

a
↓
c

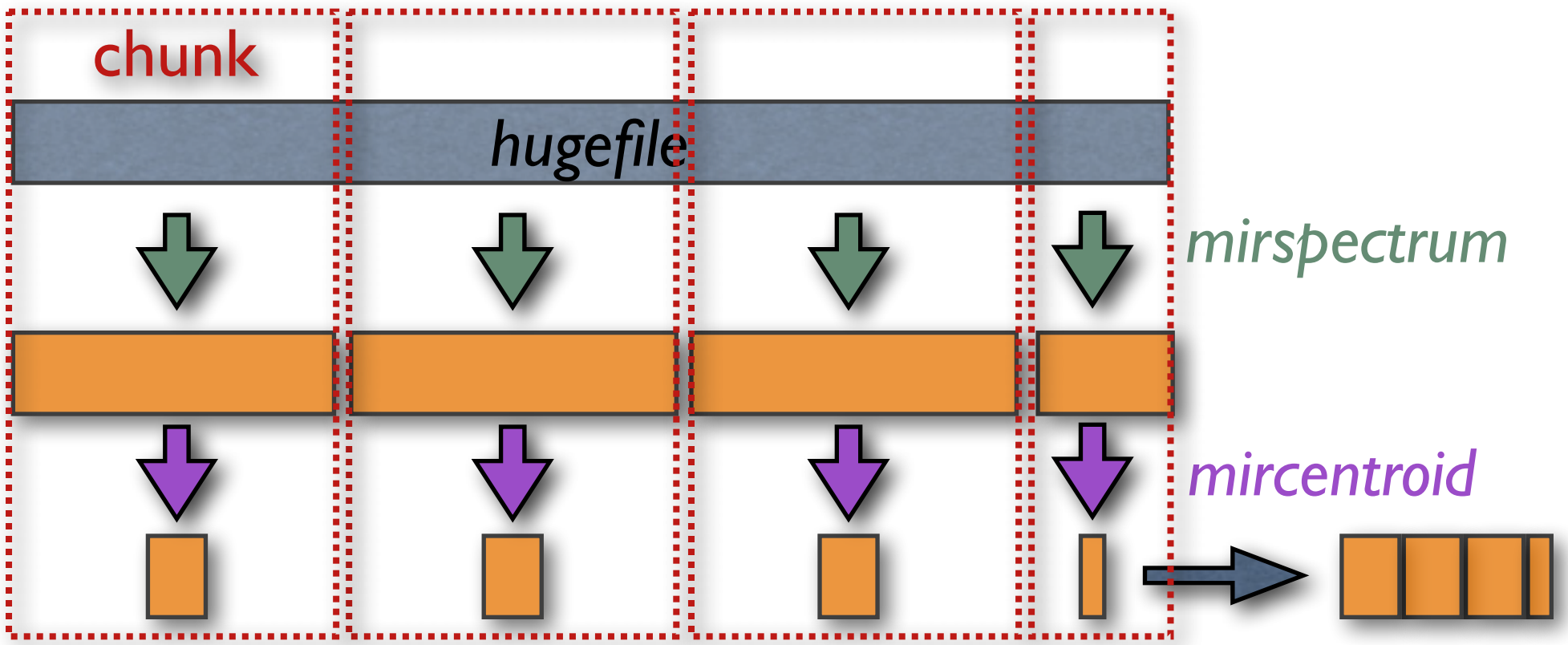


mireval

flowchart design and evaluation

- s = *mirspectrum*(‘**Design**’, ‘*Frame*’);
- c = *mircentroid*(s);
- ***mireval***(c, ‘***hugefile***’)

s
↓
c

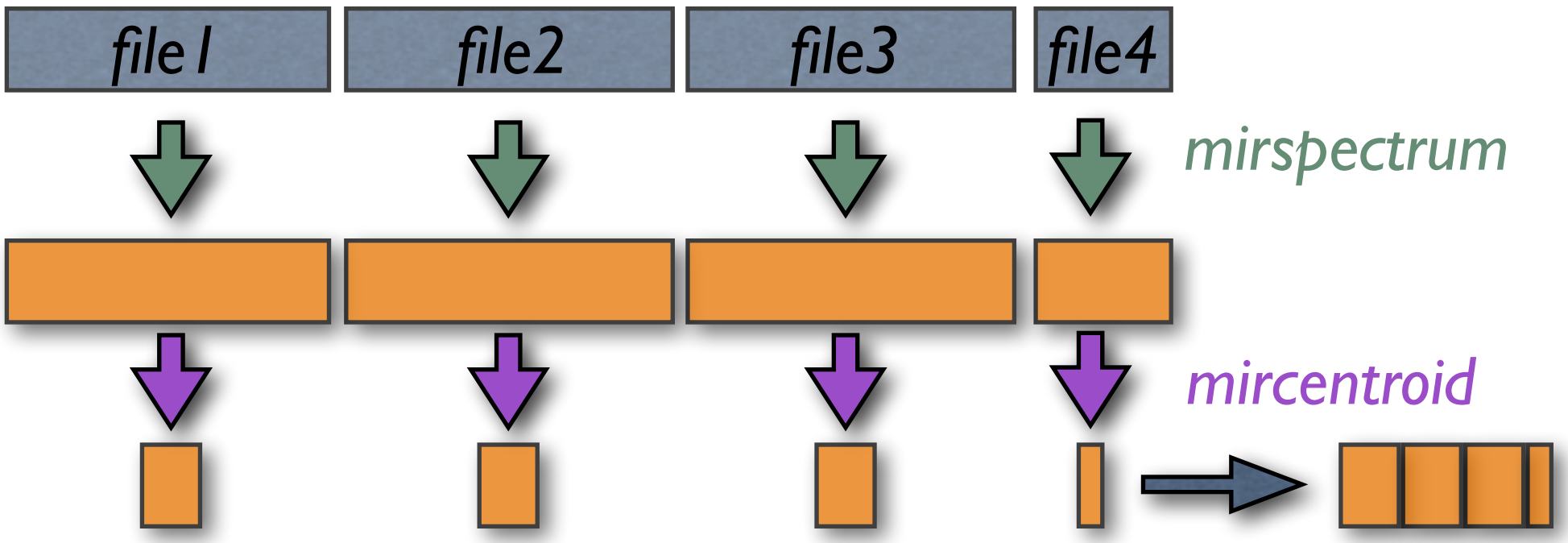


mireval

flowchart design and evaluation

- *s = mirspectrum('**Design**', 'Frame');*
- *c = mircentroid(s);*
- ***mireval**(c, '**Folder**')*

s
↓
c

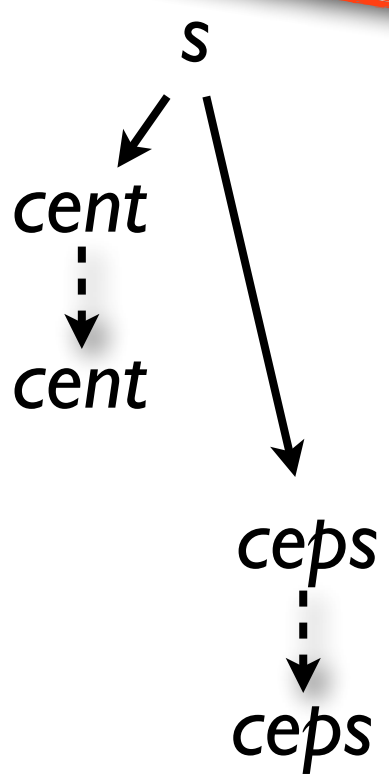


*mir**eval***

flowchart evaluation?

- *s = mirspectrum('Design', 'Frame');*
- *cent = mircentroid(s);*
- *cent = **mireval**(cent, 'Folder');*
- *ceps = mircepstrum(s);*
- *ceps = **mireval**(ceps, 'Folder');*

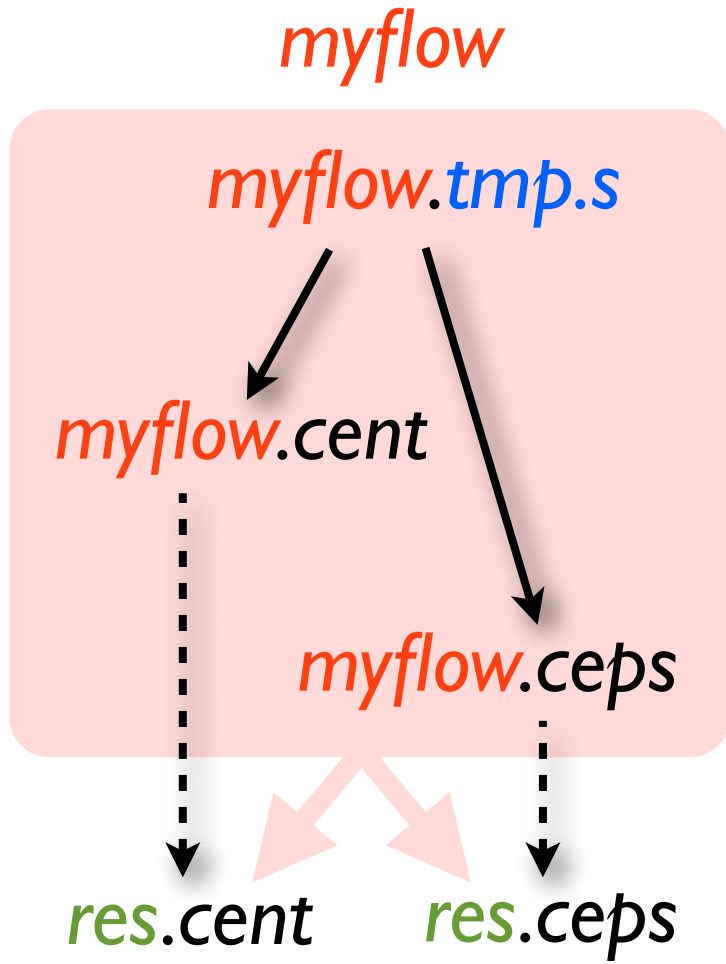
s is evaluated twice!



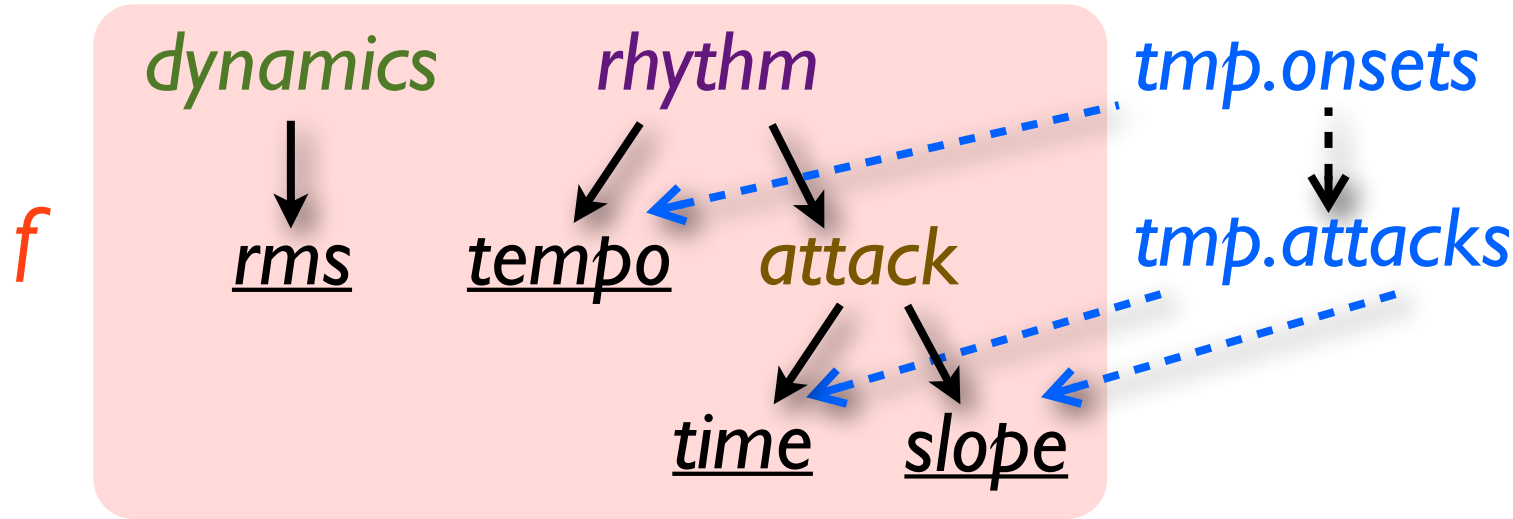
mirstruct

complex flowchart

- *myflow* = ***mirstruct***;
- *myflow.**tmp**.s* = *mirspectrum('Design', 'Frame');*
- *myflow.cent* = *mircentroid(myflow.tmp.s);*
- *myflow.ceps* = *mircepstrum(myflow.tmp.s);*
- *res* = ***mireval***(*myflow, 'Folder');*



complex flowchart



- *f = mirstruct;*
- *f.dynamics.rms = mirrms('Design','Frame')*
- *f.tmp.onsets = mironsets('Design');*
- *f.rhythm.tempo = mirtempo(f.tmp.onsets,'Frame');*
- *f.tmp.attacks = mironsets(f.tmp.onsets,'Attacks');*
- *f.rhythm.attack.time = mirattacktime(f.tmp.attacks);*
- *f.rhythm.attack.slope = mirattackslope(f.tmp.attacks);*