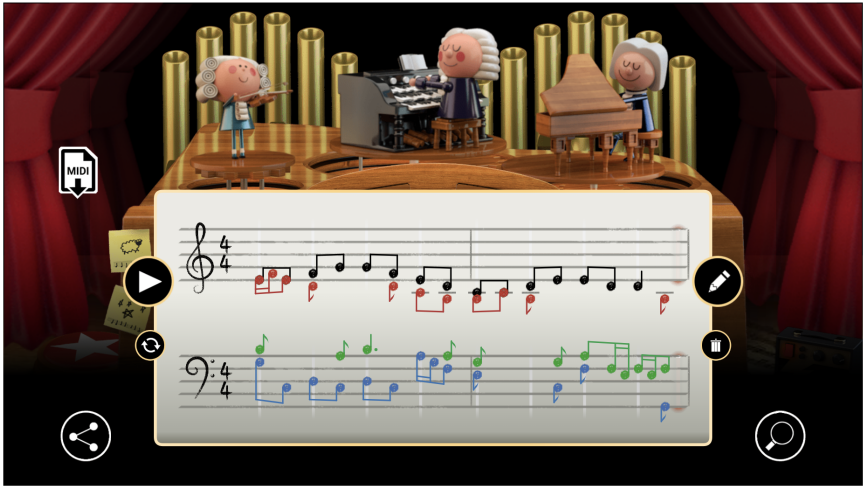
Music and Computation



What is Computer Science About?

- specify a problem for a computer
- find algorithmic solution: prescribed set of instructions for computer or person to follow
- solves any instance of that problem class
- example: given scores, check if some pitch sequence is present
- use same program for any set of inputted scores, so long as they are represented the same way

CS Research

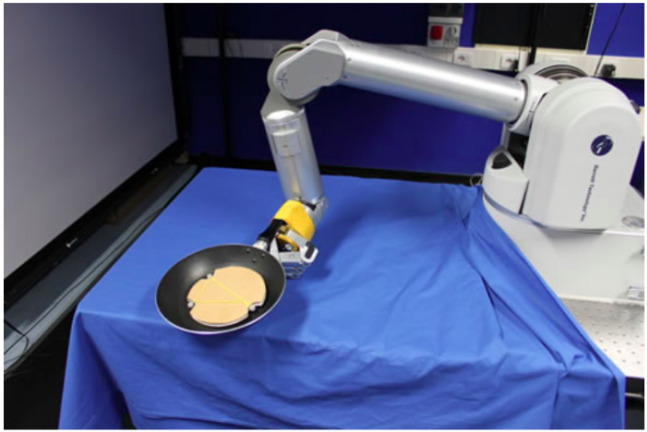
- theoretical CS: does a given problem admit an algorithmic solution? if so, how complicated?
- human-computer interaction (HCI): is this intuitive to use? can computers facilitate novel kinds of interaction?
- AI / machine learning: can a computer learn behavior from a set of examples without explicitly being programmed?

Computer Science Research (HCI)

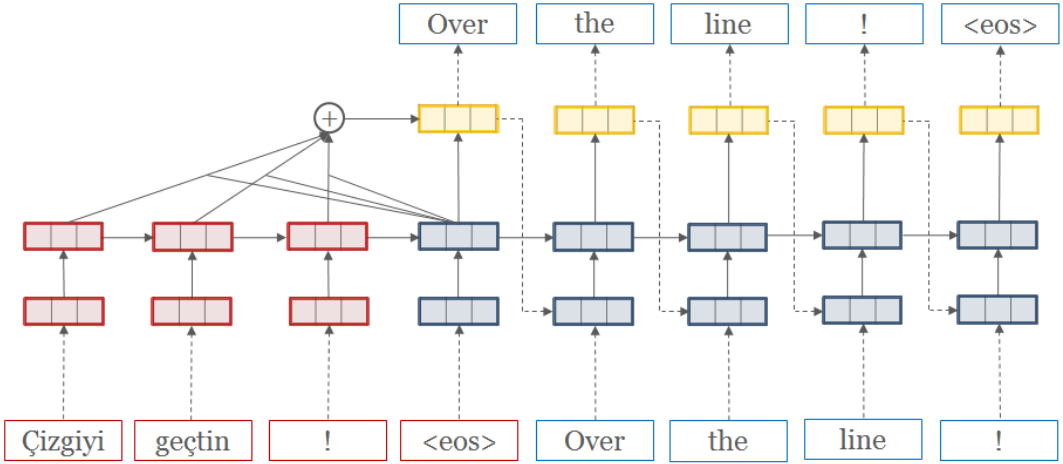
- e.g. memory assistance, collaborative ideation

Figure 1. An idea map generated by our system, showing emergent clus-

Computer Science Research: Machine Learning / AI (1)

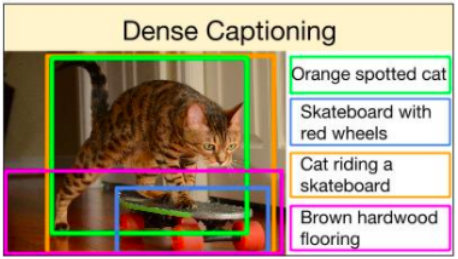


Computer Science Research: Machine Learning / AI (2)



CS in Practice

- research is interesting, what about every day user of CS?
- general means of solving daily problems: rename `IMG_1958436.jpg` automatically based on location or content?



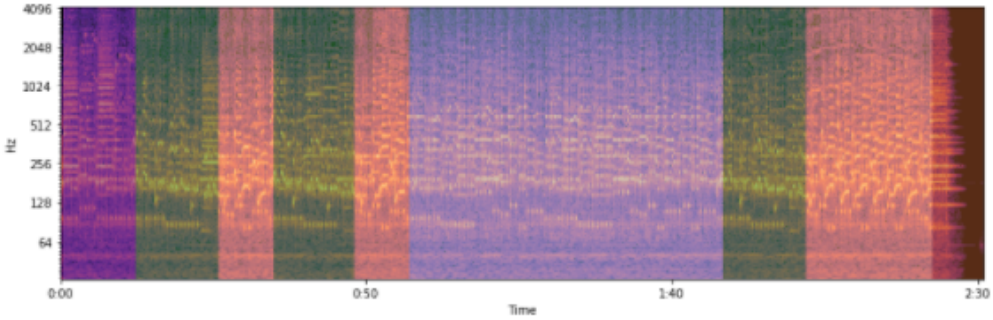
- use it in our specific field: music (examples in a few slides)

What is programming?

- “programming": act of writing such intentions in a computer language (e.g. Python)
- programming is to CS what “reading notes" is to music
- we don’t like when people ask "Oh you are a musician, so you can read notes?"
- but..., we do need to learn *some* language, and Python is a broadly applicable, popular, and easy language to use

Music and Computation: Applications (1)

- Segmentation: using frequency content to group sections of music together



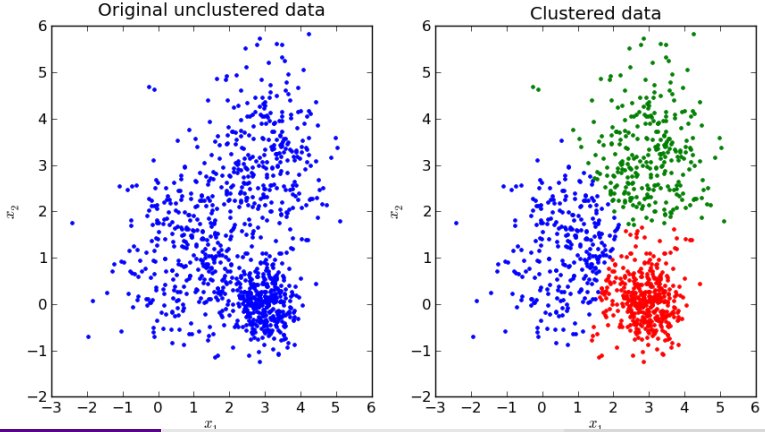
Music and Computation: Applications (2)

- music completion

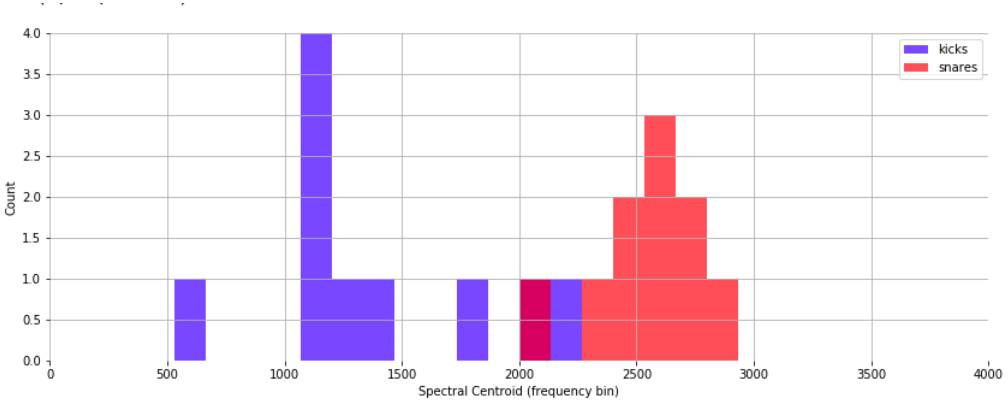
  Coconet takes incomplete musical scores and fills in the missing material. To train it, we take an example from the Bach chorales dataset of four-part counterpoint, randomly erase some notes, and ask the model to reconstruct the erased notes. The difference between Bach's composition and Coconet's fabrication gives us a learning signal by which we can train our model.

Music and Computation: Applications (3)

- clustering sounds: suppose each point is a sound and the two axes correspond to two descriptors of that sound
- group together sounds for analysis or composition

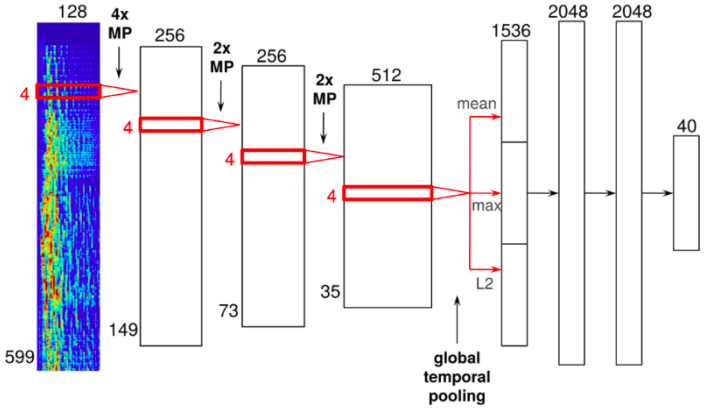


Music and Computation: Applications (3 continued)

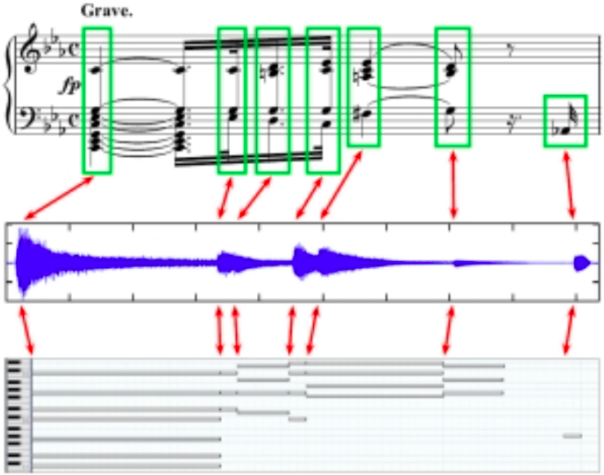


Music and Computation: Applications (4)

- does Spotify use the content of the music itself to recommend? or just mutual play counts, geographic info, etc...

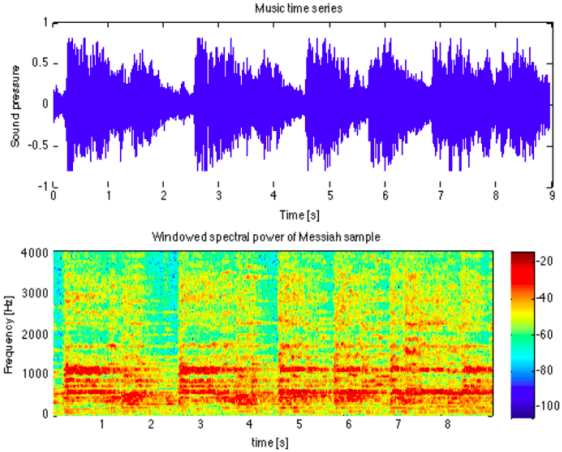


Representing Music (1)



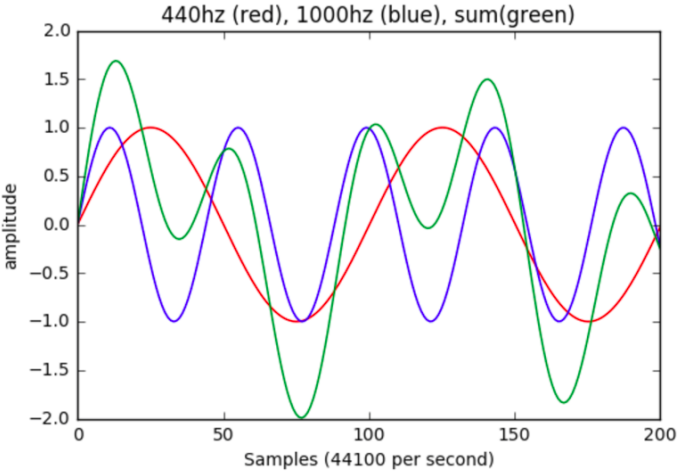
Score (top) | Audio Signal Waveform (middle) | MIDI (bottom)

Representing Music (2)



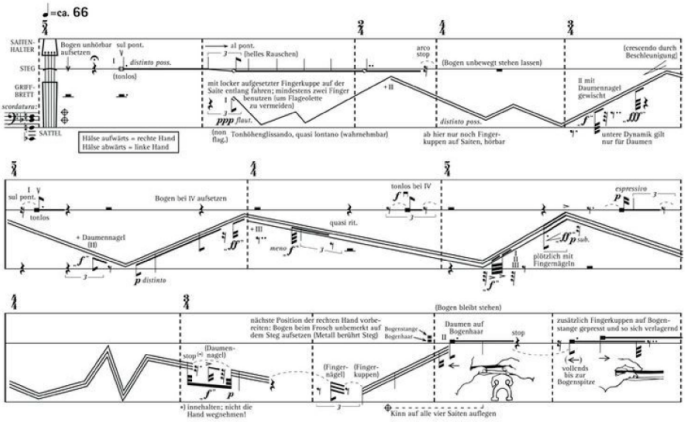
Spectrogram: decomposition of a signal into its frequency components

Representing Music (3)



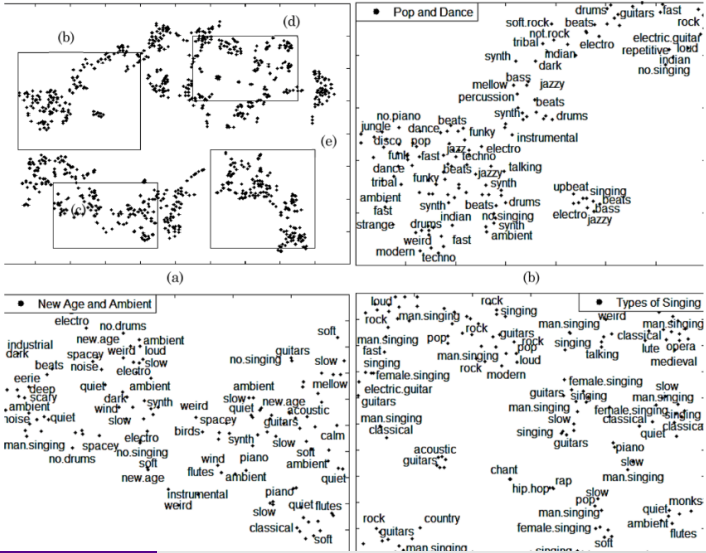
Two separate sine waves (red and blue) and their sum heard as a chord (green)

Representing Music (4)



Pression for Solo Cello by Helmut Lachenmann

Representing Music (5)



Music and Computation: Music 21

With five lines of music21 code or less, you can:

Prepare a thematic (incipit) catalog of every Bach chorale that is in 3/4:

```
catalog = stream.Opus()
for workName in corpus.chorales.Iterator():
  work = converter.parse(workName)
  firstTS = work.recurse().getElementsByClass('TimeSignature')[0]
  if firstTS.ratioString == '6/8':
    catalog.append(work.measures(0, 2))
catalog.show()
```

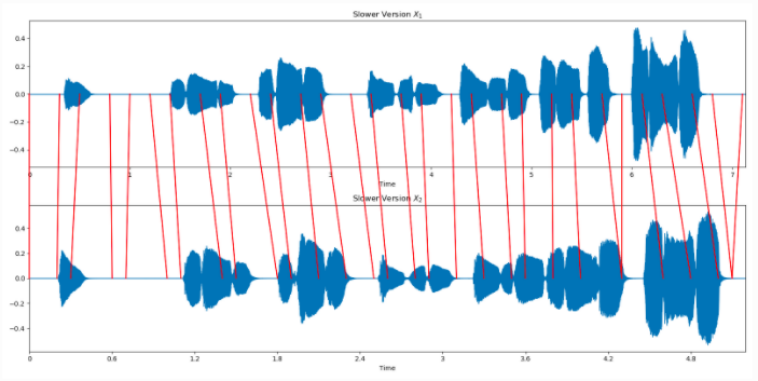
Google every motet in your database that includes the word 'exultavit' in the superius (soprano) part (even if broken up as multiple syllables in the source file) to see how common the motet's text is (assuming you have a bunch of motets in "listOfMotets"):

```
import webbrowser
for motet in listOfMotets:
  superius = motet[0]
  lyrics = text.assembleLyrics(part)
  if 'exultavit' in lyrics:
    webbrowser.open('http://www.google.com/search?&q=' + lyrics)
```

Add the German name (i.e., B♭ = B, B = H, A♯ = Ais) under each note of a Bach chorale and

Music and Computation: Librosa

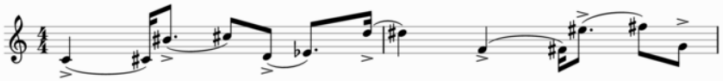
- dynamic time warping: find correspondence in audio of two different performance of same music



Music and Computation: Abjad (1)

Now, let's split the notes you just made every 5/16 duration, transpose every other split group up by a major-seventh, then slur every split group and finally attach an accent to the first note of each split group:

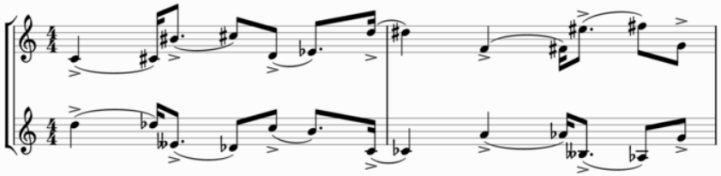
```
>>> shards = abjad.mutate(staff[:]).split(
...     durations=[abjad.Duration(5, 16)],
...     cyclic=True,
...     tie_split_notes=False,
...     )
>>> for index, shard in enumerate(shards):
...     if index % 2:
...         abjad.mutate(shard).transpose('M7')
...     if 1 < len(shard):
...         abjad.attach(abjad.Slur(), shard)
...     abjad.attach(abjad.Articulation('accent'), shard[0])
...
>>> abjad.show(staff)
```



Music and Computation: Abjad (2)

Now let's create a second staff, copied from the first, invert all of the new staff's pitches around middle-G and finally group both staves into a staff group:

```
>>>
>>> copied_staff = abjad.mutate(staff).copy()
>>> staff_group = abjad.StaffGroup([staff, copied_staff])
>>> for note in abjad.iterate(copied_staff).leaves(pitched=True):
...     note.written_pitch = note.written_pitch.invert(axis='G4')
...
>>> abjad.show(staff_group)
```



So What Will We Learn?

- how to program in Python
- how music is represented digitally
- uses of python in computer-assisted musicology/theory/composition/more
- some (not *too* much) relevant math if its necessary

and

- discuss how these skills can be useful for everyone's upcoming research projects / compositions / analyses