JAZZ SOLO ANALYSIS BETWEEN MUSIC INFORMATION RETRIEVAL, MUSIC PSYCHOLOGY, AND JAZZ RESEARCH.

Part II – Data and Tools

Wolf-Georg Zaddach, **Klaus Frieler**, Jakob Abeßer

August 6, 2016

University of Music “Franz Liszt” Weimar, Germany
Fraunhofer IDMT, Ilmenau, Germany

Outline

1. Introduction
2. Data Model
3. (Analysis) Tools
4. Summary & Outlook

Introduction

Introduction

The Jazzomat Research Project

- Aims:
  - Computational & statistical jazz research.
  - Studying the creative processes during jazz improvisation.
  - Style comparison, jazz history, jazz theory.
- Work packages:
  - Building a representative database of high-quality jazz solo transcriptions (Weimar Jazz Database, WJazzD).
  - Development of symbolic analysis tools (MeloSpyLib/Suite/GUI).
  - Development of score-informed audio analysis techniques (cf. Part III of this tutorial).

Funded by DFG (German Research Foundation), “Melodisch-rhythmische Gestaltung von Jazzimprovisationen. Rechnerbasierte Musikanalyse einstimmiger Jazzsoli” (DFG-PF 669/7-1). Oct 2012 – Mar 2017 (4.5y)

Motivation

- Why **yet another toolkit** for symbolic music analysis?
- Answer: None of the existing toolkits met our demands.
- Nearly all available software is based on **notation-oriented** data models (e.g., `humdrum`, `music21`).
- Our approach: **Performance-oriented** monophonic data.
- Why only monophonic data?
  - Because it's **easier**.
  - **Monophonic** solo instruments actually **central to jazz** → Reasonable constraint.

Performance-oriented data

- Based-on descriptions of **actually sounding tones**.
- Note-level annotations added later (anyway not unique).
- NB: For many MIR problems, there is actually more **"shaky"** than **ground truth**.
- Listener's or performer's perspective? (But every human performer is also a listener.)

Music Representations

- Use one (of potentially many) **psychologically plausible** interpretations of actual acoustic events.
- Assumption: There is a **sufficiently well-defined mathematical representation** of musical tone events which captures important aspects (common to listener's and performer's perspective).
- Can be (partly) **validated by re-synthesis**.
- Due to precision of human perception and inter-subjective variation, re-presentations are actually **probability distributions**.
- Music representations are actually **psychological models** (or instruction sets).

Data model

Data Model

- Our data model is based on **tone events**.
- (Exercise: Discuss the distinction between tones and notes.)
- Assumptions:
  - A tone event has defined **onset**, **duration** and **pitch**.
  - A solo S is a list of tone events m_i, re-presented with tuplets

$$m_i = (t_i, d_i, p_i).$$

  - Tone events and solos are enriched by **annotations** $\vec{a_i}$:

$$m_i = (t_i, d_i, p_i, \vec{a_i}).$$

Data Model

Annotation types

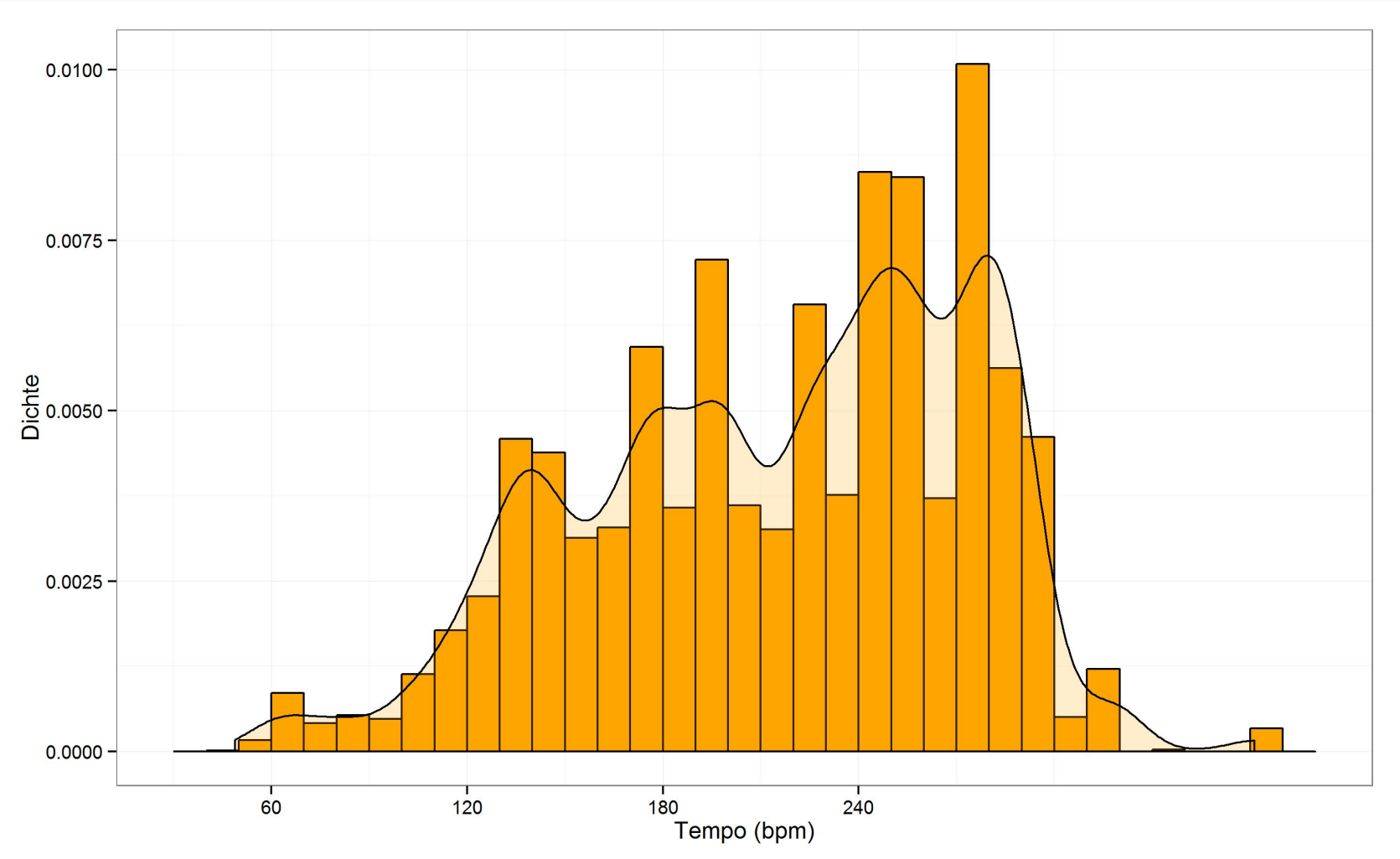
- Global annotations:
  - Metadata (manual)
  - (Annotated) beat tracks (manual, transcriber)
- Sectional annotations:
  - Phrases (manual, transcriber)
  - Chord context (manual, from lead sheets)
  - Form parts (manual, from lead sheets)
  - Mid-level units (manual, transcriber)
- Local annotations:
  - Frequency modulations (manual, transcriber, see next part)
  - Metrical annotations (semi-automatic, based on beat track)
  - intensities (automatic, see next part of tutorial)

Data Model: Global Annotations

Solo info

- Performer
- Title, title add-on
- Solo part
- Instrument
- Style (traditional, swing, bebop, cool, hard bop post-bop, free)
- Avg. tempo (bpm)
- Rhythm feel (twobeat, swing, funk/rock, latin, mixed)
- Key (major, minor, blues, modes)
- Time signature (global)
- Chord changes
- Number of choruses
- Track reference

Example: Tempo distribution



Data Model: Global Annotations

Track info

- File name track
- Record reference
- Line-up (single string)
- MusicBrainzID (of track)
- Number of track in record
- Recording date
- Composition reference

Record info

- Artist
- Record title
- Record label
- Record number
- MusicBrainzID (record)
- Release date

Data Model: Global Annotations

Transcription info

- File name SV project
- File name of solo cut
- Solo time (begin:end, low precision)
- Start time of solo in track (high precision, automatically extracted)
- Transcription status (PREFINAL, FINAL)
- Track reference

Composition info

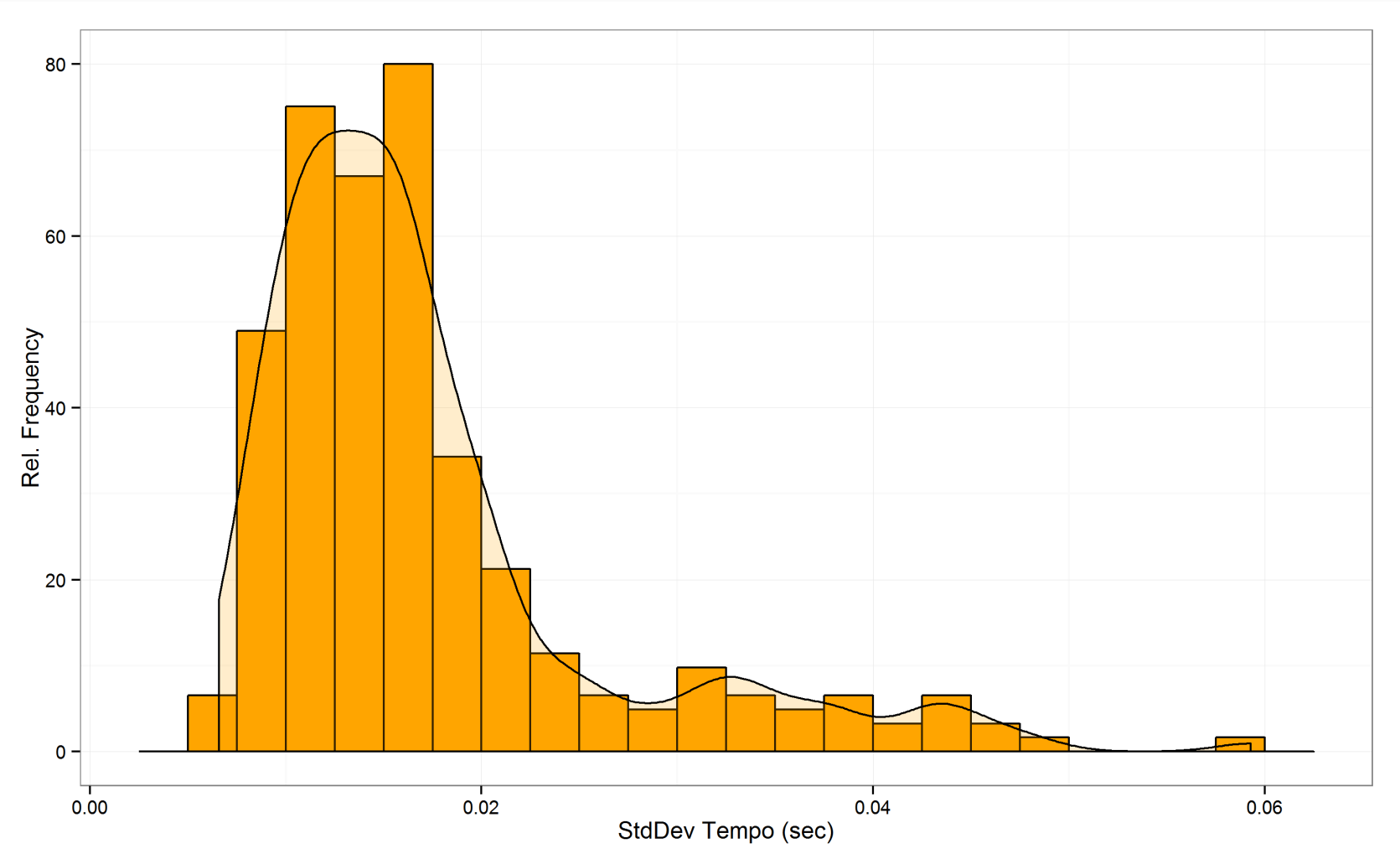
- Title
- Composer
- Form (AABA etc.)
- Tonality type (functional, blues, modal, color, free)

Data Model: Global Annotations

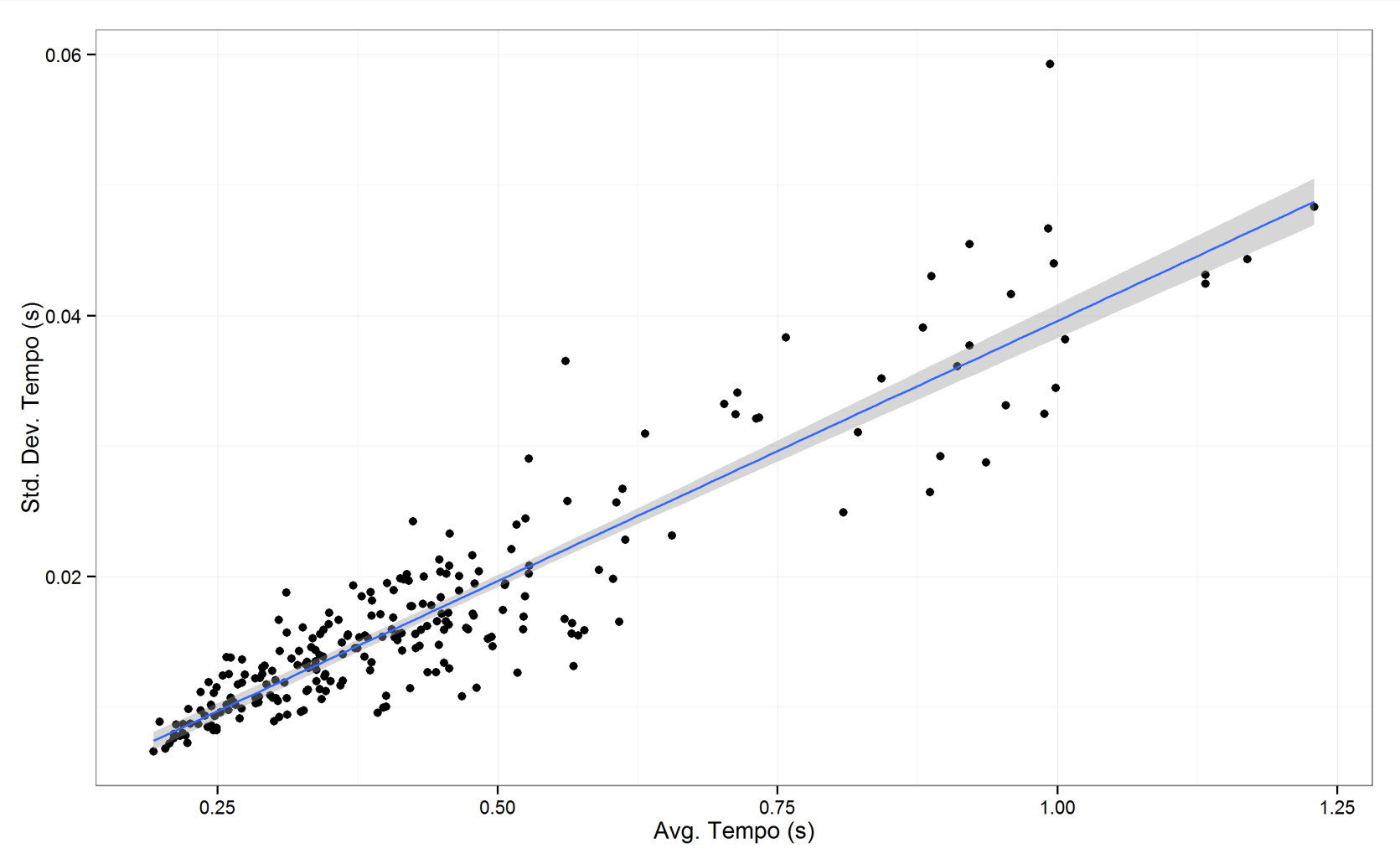
Annotated beat tracks

- Beats manually tapped & cross-checked.
- Includes half-time tapping option for very fast tunes (every second beat interpolated).
- Chord, form, and signature information are tagged to beat events.
- Chords and form taken from lead sheets ($\Rightarrow$ only restricted use as ground truth).
- Contains beat-wise bass pitches (experimental).

Example: Standard deviation beats



Example: Tempo vs. Standard deviation beats



Data Model: Sectional Annotations

Phrases

- Phrases are **basic perceptional units** (melo-rhythmic gestalts).
- Problem: Large **inter-subjective variability**, but only one annotator.
- Remedies:
  - In jazz solos often rather clear units (breathing pauses).
  - Annotate only the clearest/largest units.
  - Mid-level analysis provides **second phrase annotation**.

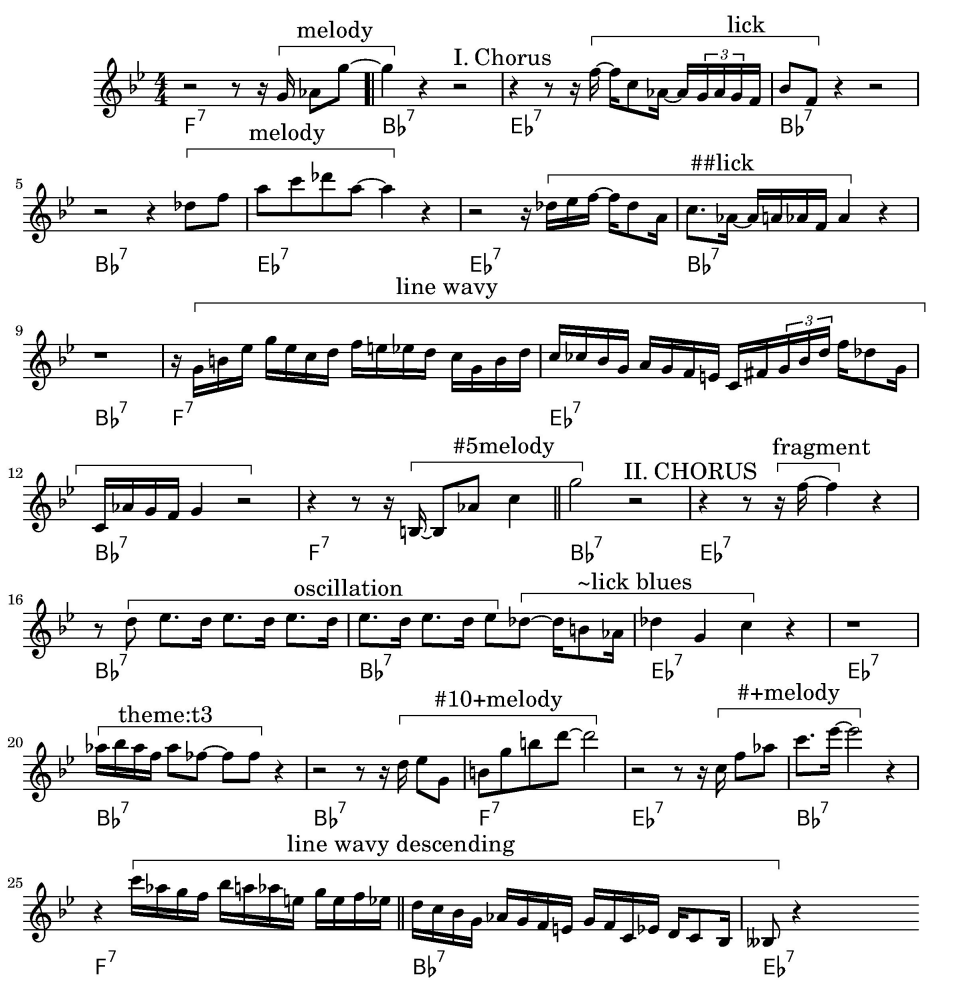
Data Model: Sectional Annotations

Mid-level Analysis

- Newly developed analysis method (Frieler et al., 2016)
- Categorical system of 9 main types of playing ideas: `line`, `lick`, `rhythm`, `melody`, `void`, `theme`, `quote`, `expressive`, plus several sub-classes and sub-sub-classes.
- Manual annotation, but high inter-rater agreement for mid-level boundaries, medium agreement for category labels.

Example: Mid-level Analysis

Sonny Rollins – Blue Seven



Data Model: Local Annotations

- Frequency Modulations
  - Event-tagged annotation of frequency modulations.
  - Classes: `vibrato`, `slide`, `bend`, `fall-off`.
- Intensities
  - Median,
  - maximum,
  - standard deviation,
  - relative peak position.
  - Centroid

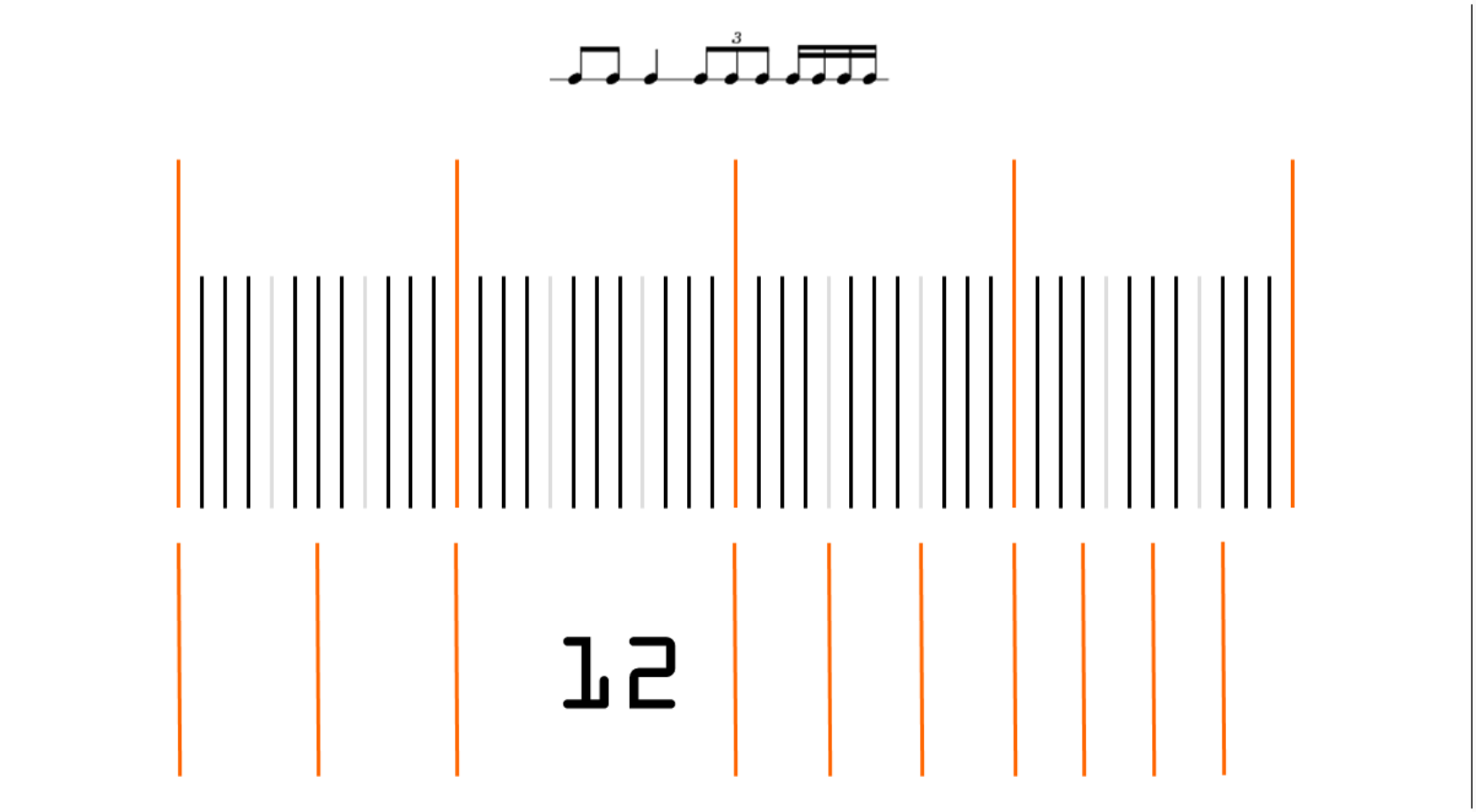
→ See next part of tutorial!

Data Model: Local Annotations

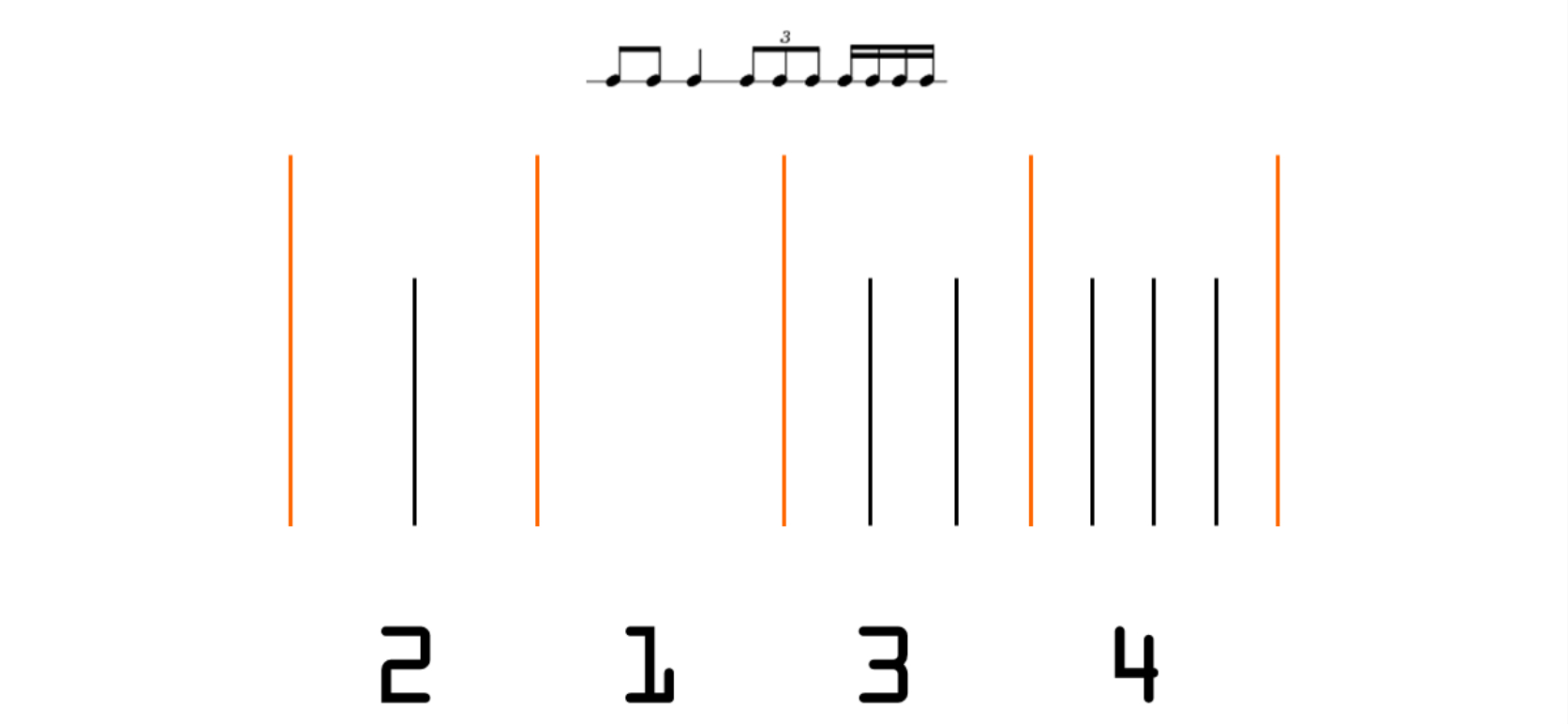
Metrical Annotation

- Meter is a **fundamental** aspect of music.
- Interesting for analysis, necessary for score creation.
- Metrical annotation done via **FlexQ algorithm** based on beat tracks and tone onsets.

Grid Quantization



Flexible Quantization (FlexQ)



FlexQ Algorithm

- Input:
  - Meter-annotated beat track.
  - Tone onsets.
- Goal:
  - Find **optimal beat-wise grid** for given onsets.
  - Derive metrical annotations.
- Method:
  - Simple **preference rule-based** optimization.
  - Test **all possible grids** (up to a certain limit) with a penalty function.
  - Constraint: Strictly monophonic (keep onset order).

FlexQ Algorithm

Preference Rule system

- Prefer fully occupied grids.
- Prefer smaller onset-to-grid-distances (quantization error).
- Prefer even-numbered grids.
- Prefer homogeneous shifts (minimize SD of quantization error).

FlexQ Algorithm

- Evaluation
  - No large scale evaluation yet due to **hard-to-get-by** ground truth.
  - But it **works seemingly fine** … (by indirect and informal metrics). See: Database Content
- Known issues
  - Beat range has to be extended **slightly before** the first beat for common anticipations (playing ahead).
  - Slow tempo ($<$ 80bpm) results often in very large divisions.
  - Trade-off between simplicity and accuracy.
  - **slides** and **glissandi** resolved into very short events, should be better captured as **out of meter** (appogiaturas).

Data: Local annotations

Metrical annotation

- Each event has a **metrical position** and a **metrical context**, containing meter and beat information.
- Meter information:
  - **Number of beats** in the current bar (period P).
  - (Idealized) **beat proportions** for non-isochronous beats (e.g., 3+2+2).
  - Classical **signature**.
- Beat information:
  - **Duration** of current beat interval.
  - **Division** D (number of tatums).
  - **Tatum proportions**.

Metrical annotation

Metrical position syntax

- Bar number ($\in \mathbb{Z}$),
- beat position $\in [1 : P]$,
- tatum position $\in [1 : D]$,
- subtatum (currently unused).
- Short annotation syntax (for output):

```
<period>.<division>.<bar>.<beat>.<tatum>[.<subtatum>]
```

- Example: `4.2.5.1.1`: First beat in fifth bar in a 4/4 with binary division of the beat.

Tools

Tools

Overview

- Freely available binaries:
  - **MeloSpySuite** (batch processing, extended options)
    - `melconv`
    - `melfeature`
    - `melpat`
  - **MeloSpyGUI**
    - QT4-based GUI for MeloSpySuite functionalities.
    - Simple visualizations (piano-roll, bar plots, scatter plots).
  - Unreleased binary `melbundle` for SV project file checking.
  - Based on **yet unpublished** Python (2.7) library **MeloSpyLib**.

Tools: `melconv`

Overview

- File import/export.
- Strictly monophonic.
- Adds as much missing information as possible (e.g., metrical annotation).

Tools: `melconv`

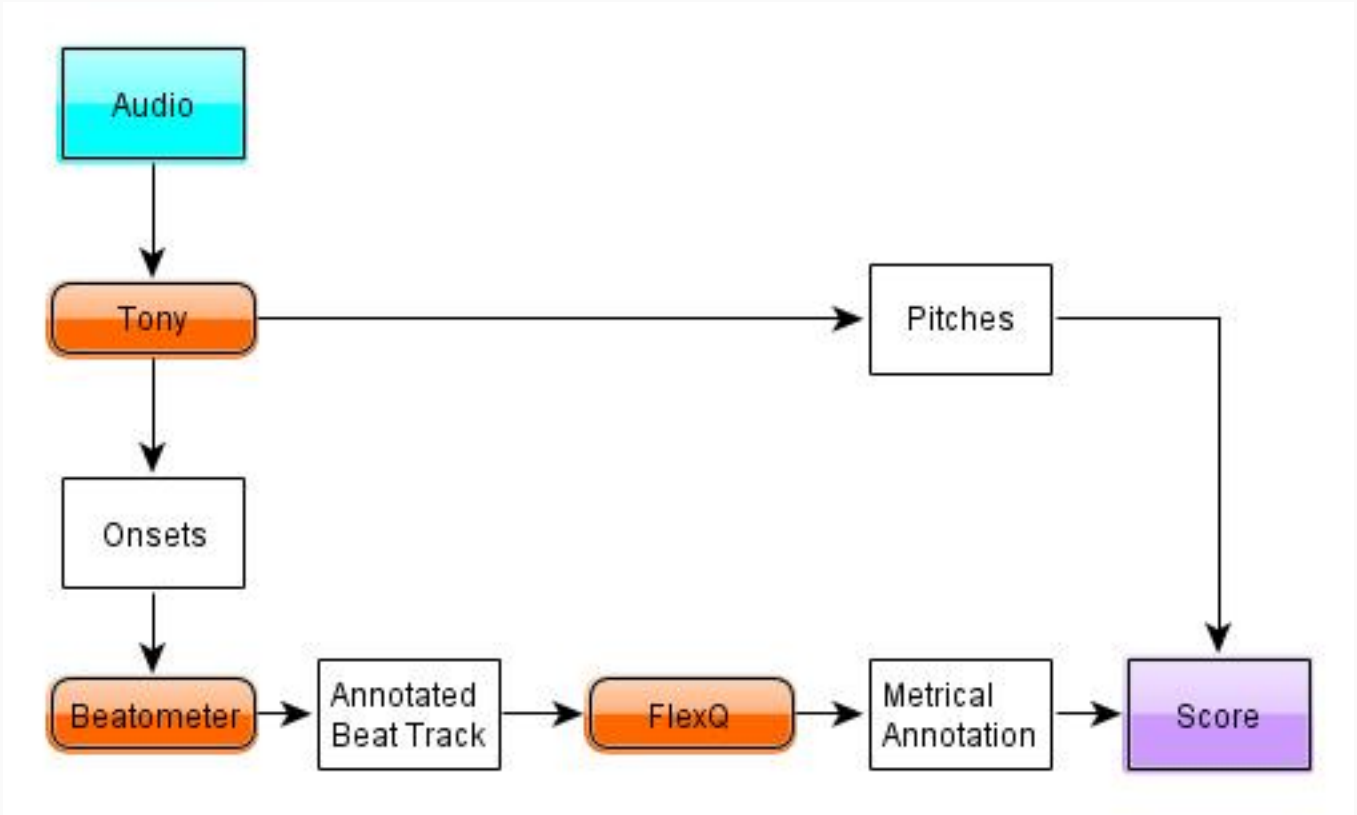
Input formats:

- Jazzomat style SV project files (cf. first part)
- WJazzD SQlite3 DB
- MIDI
- EsAC
- MCSV (old custom CSV format)
- Tony/pYIN note tracks
- `**kern` (removed in new release)

Output formats:

- WJazzD SQLite3 DB
- MIDI
- Lilypond
- MCSV
- MCSV2 (new custom CSV format)

Example: Tony Note Track → Lilypond



Tools: melfeature

Overview

- Feature extraction tool (scalar, vector and matrix features).
- **Modular** feature scripting engine (“Feature Machine”).
- Processing chains defined in **Feature Definition Files** (YAML).
- Allows easy **extension**, **modifications** and **configuration** of features without changing the code base as well as automated documentation.
- Currently, **95 predefined FDFs** with **636 features** (partly overlapping).

Tools: `melfeature`

Feature machine

- Features defined by a **source**, an arbitrary number of **process modules** and a **sink**.
- Sources are basic **features exports** from the library (about 100 incl. 60 structural markers).
- About 20 available processing modules.

Tools: melfeature

Source features

1. **Metadata** (e.g., performer, title, key, tonality type).
2. **Structural annotations** (e.g., chord context, phrase, form part & chorus IDs, metrical position).
3. **Abstractions** (transformations, viewpoints) (e.g., pitch, pitch class, interval, IOI, duration classes).

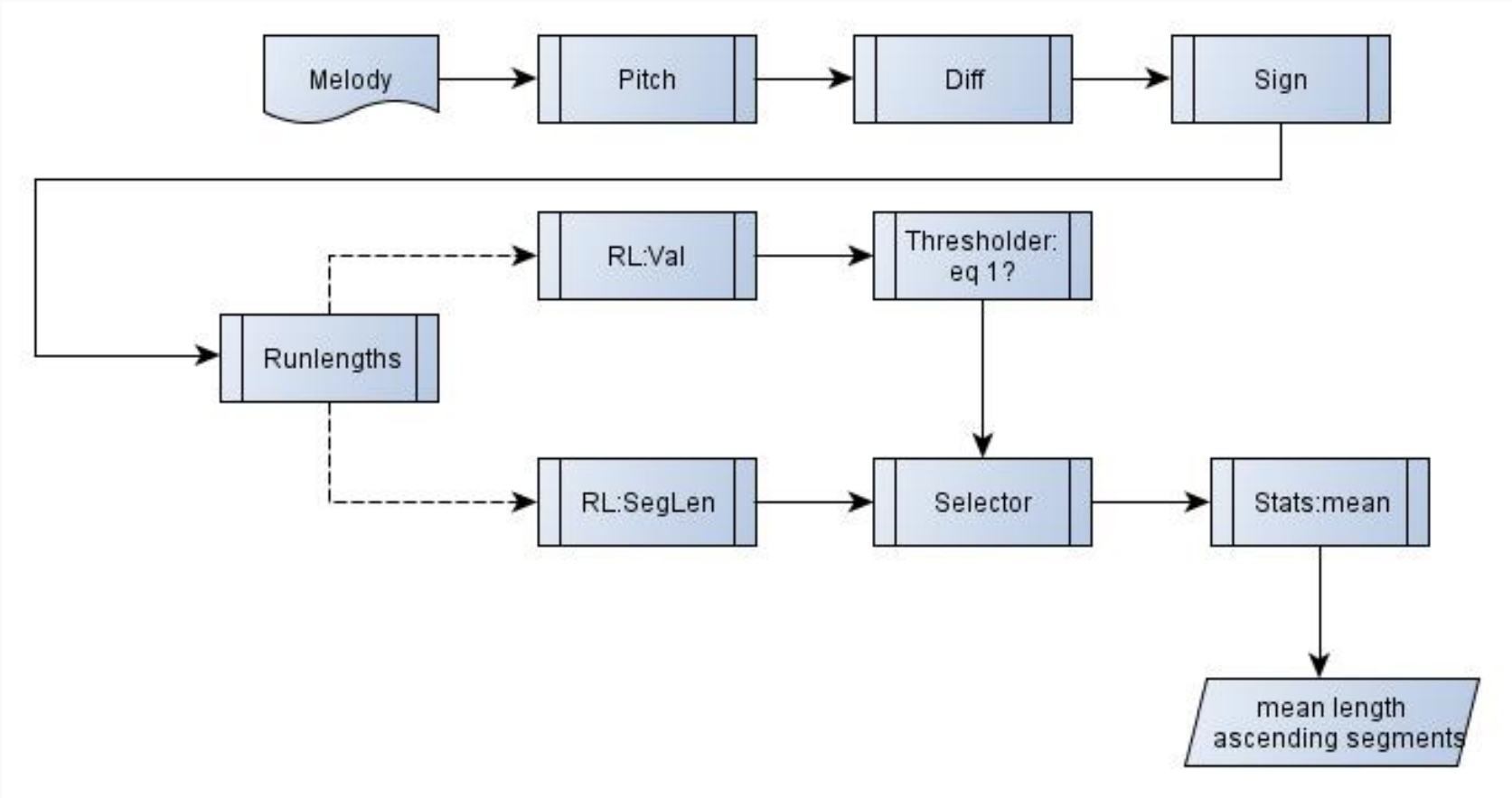
Tools: melfeature

Processing modules types

- Structural: append, cartProd, selector, truncate, unique ...
- Calculation: arithmetic (+, -, /, *), abs, sum, mod, logic ...
- Statistics: mean, median, standard deviation ...
- Auxiliary: index, length, thresholder...
- Pattern: ngrams, markov ...
- Special: selfSimilarity, runLength ...

Example: Feature Process Chain

Feature: Mean length of ascending segments



Tools: melfeature

Feature categories

- Accents (structural marker)
- Auxiliary
- Contour
- Intervals
- Metadata
- Meter
- Mid-level Analyis
- Pitch
- Rhythm
- Sequence features (pitch, interval, rhythm)
- Tone formation (timbre)

Tools: melfeature

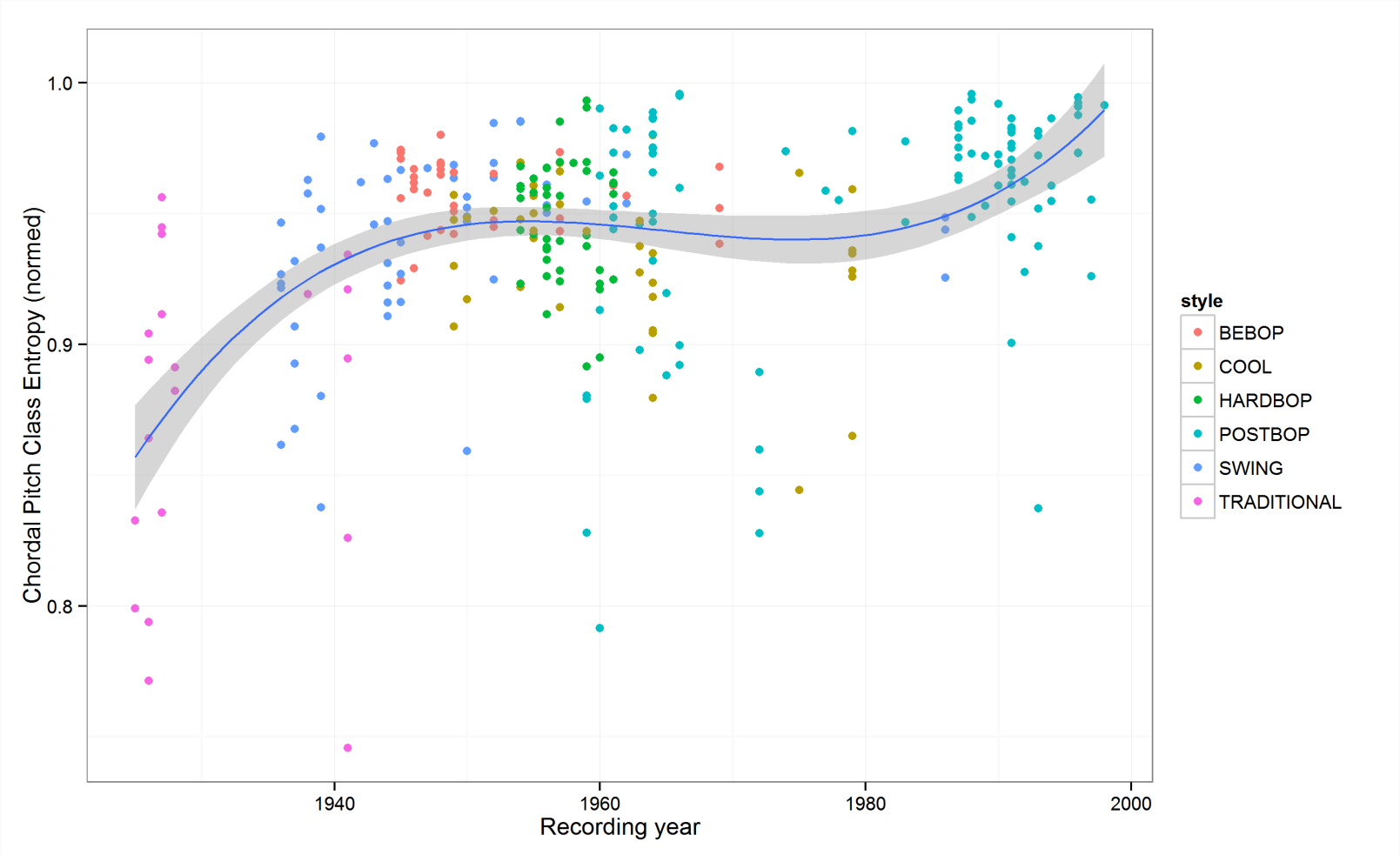
Segmentation

- Features can be calculated for **entire melodies** or **segments**.
- Available segments (WJazzD): Phrases, choruses, form parts, bars, note chunks, mid-level units).

Output

- Output as CSV files in long (default) or wide format.
- Possible to combine scalar and vector feature in long format.
- Matrix features (e.g., self-similarity of phrases) only in wide format.

Example: Evolution of chordal pitch class entropy



Tools: melpat

Overview

- Tool for **pattern mining** and **search**.
- Patterns central for jazz creativity research.
- Available for **predefined set** of abstractions.
- Three operational modes:
  - Two-stage pattern search with regular expressions.
  - Pattern partitions with respect to a corpus.
  - Raw n-gram distributions.

Tools: melpat

Available abstractions

- Interval, fuzzy interval, parsons.
- Pitch, pitch class, [extended] chordal [diatonic] pitch class, tonal [diatonic] pitch class.
- Duration classes, IOI classes (absolute/relative)
- Metrical circle map (48), metrical weights.
- Accents/structural marker.

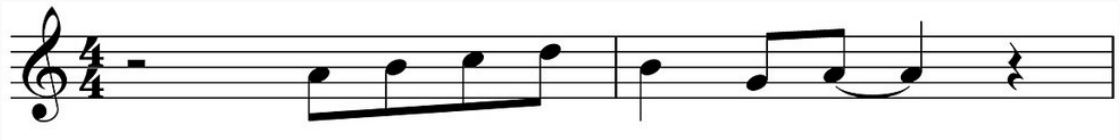
Tools: melpat

Pattern search

- Two-stage Python-style **regular expression search**.
- Implemented by mapping abstraction alphabets to (arbitrary) unicode characters.
- **Secondary search**: Search in result set of primary search (mostly different abstraction).
- Output formats: List of all occurrences, pattern statistics, or MIDI.

Example: Pattern search

- Search for“The Lick”.



- Search pattern:

```
int: [+2, +1, +2, -3, -4, + 2]
```

- Results:

id	start	N	onset	duration	metrical position	freq
Chet Baker: Let’s Get Lost	2	6	3.23	2.09	4.2.1.3.1	2
Woody Shaw: Dat Dere	280	6	96.55	1.13	4.4.44.2.3	2

Tools: melpat

Pattern partitions

- A pattern partition finds all n-grams in a melody subject to certain conditions in relation to a corpus of melodies.
- Conditions:
  - Minimum and maximum n-gram lengths.
  - Minimum n-gram frequency.
  - Minimum number of different sources.
- True sub-patterns are filtered.
- Special options for filtering scales, arpeggios and trills.
- Can also be carried out on Markov-simulated corpora for comparison.
- Output formats: List of all occurrences, statistics.

Example: Pattern partition

- Find Chordal Diatonic Pitch Class partitions with patterns of minimum length 4 occurring in all Charlie Parker solos in the database.
- Results:
  - Only two patterns: **6543** (32), **5432** (12)
  - Pattern coverage: from .045 (“Thriving on a Riff”) to .156 (“Scrapple from the Apple”)
  - Coverage: Percentage of all solo tones contained in the partition patterns.
- Further statistics: over-coverage, avg. overlap, avg. N, (avg.) log of excess probabilities ($\log p_{obs}/p_{expected}$)

Tools: `melpat`

N-gram databases

- Similar to partition, but without filtering.
- Can be used for n-gram & Markov models.
- Output format: List of all n-grams with positions, statistics.
- **Future idea**: n-gram distribution exchange format & public repository.

Summary & Outlook

Summary

- Weimar Jazz Database: A high-quality database for monophonic jazz solos.
- Comprehensive information about solos (you can't ask for much more...).
- Provides ground truth for several classical MIR tasks (though standard problem of audio file distribution).
- Tools allow a vast and flexible array of analyses for musicological (and MIR) research.
- MeloSpyGUI greatly enhances UIX (for non-experts and experts).

Outlook

- Final release of Weimar Jazz Database end of 2016.
- Open-source release of **MeloSpyLib** (spring 2017) on github.
- Improvements:
  - More features, advanced features.
  - Optimized and extended pattern search output (e.g., audio/score).
  - Pattern search by sample, data selection by patterns.
  - More input & output formats, music21 interface.
- New modules:
  - `melharm` (harmonic & tonality analysis).
  - Melodic similarity.
  - Segmentation.
  - Generative models.
- Extension to polyphony & different tone systems.

End of Part II

Thank you!

Questions?

http://jazzomat.hfm-weimar.de