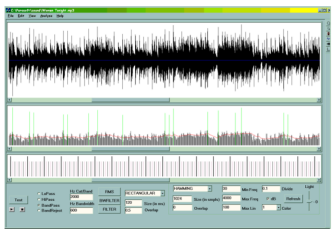
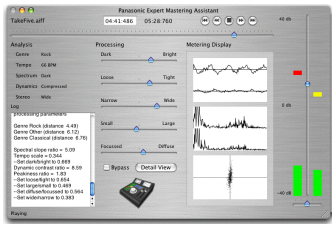
SndsLike & BirdsEarApp: MIR in Action

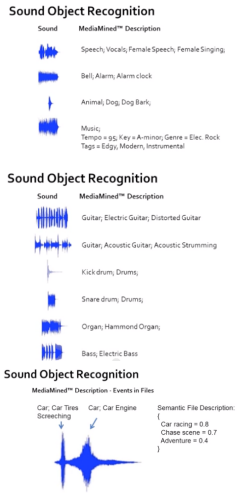
Stephen T. Pope
stephen@FASTLabInc.com
MAT 240E
Feb 24, 2014

SndsLike Introduction

- Overview
  - Similarity measures
  - Genre/artist clustering
  - Recommender systems
- Background
  - Prior art - big field(s)
  - My previous Music DB work
    - HyperScore ToolKit, MODE (1980s)
    - Paleo, OperaBrowser, Siren (1990s)
    - FMAK 1 for Predixis (MusicIP, GraceNote, OFA)
    - FMAK 2 for Expert Mastering Assistant (2002-04)
    - FMAK 3 for Catalyst search engine (2006-08)
    - FMAK 4 for Imagine Research (2010-12, iZotope)







SndsLike Goals

- Similarity-based "recommender" system aimed at production music data sets (why?)
- Written from scratch (I sold the old code to iZotope)
- Simple, fast, portable, embeddable
  - C++, octave, java, python, (My)SQL DBMS
- Use the “latest features” (> 400 features)
- Use the “latest statistics” (sophisticated de-noising)
- Use the “latest distance metrics” (learned)
- Use existing noisy/partial labels to train clustering and distance metrics

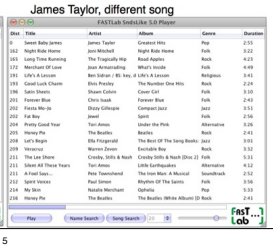
SndsLike “Demo”

Punk

Vivaldi

James Taylor

James Taylor, different song



SndsLike “Demo”

Electric Blues

Metallica

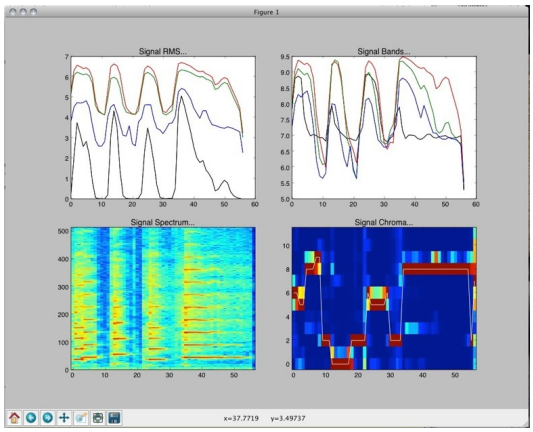
Relaxing solo piano

New-age chanting

The “Latest Features”

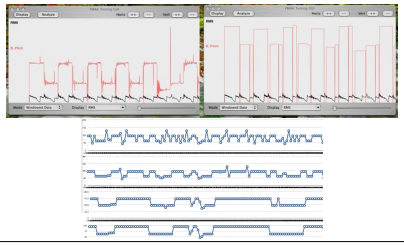
- Standard time- and freq-domain features
  - HPF/LPF versions
  - Many freq bands
- Chroma, harmonicity, MFCCs & spectral measures
  - Sp-slope, spread, bandwidth, variety, kurtosis, roll-off...
  - Spectral tracking and track birth/death stats (useful)
- Fluctuation pattern features (E Pampalk)
- Beat histograms (G Tzanetakis)
- Statistical Spectrum Descriptors (Lidy & Rauber)
- Several tempo estimates (+ stats)
- Several bass pitch estimates (+ stats) + tracking
- Several chord/key pitch estimates (+ stats)
- Musical segmentation and segment-related features

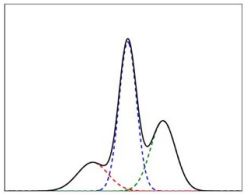
Feature Extractor Development

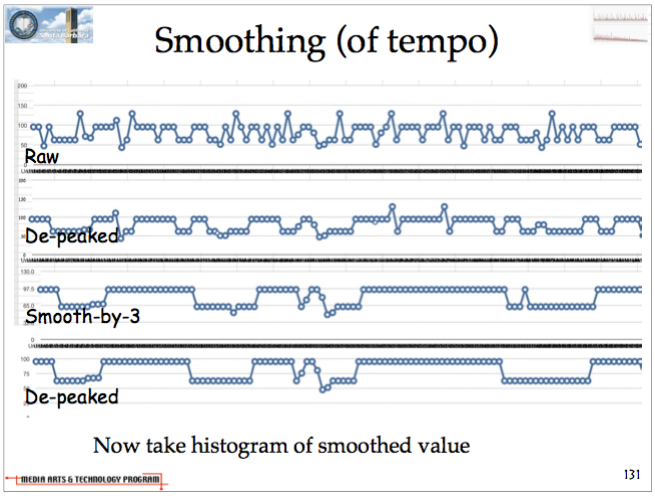


The Latest Statistics

- Lots of feature-dependent smoothing
  - Data mode: noisy, bi-modal, clicky, etc.
- Take Gaussian Mixture Models (GMM) of all features
- Save gmm-avg, main-lobe width/weight, bi-modality...
- Also save dev, del, del2







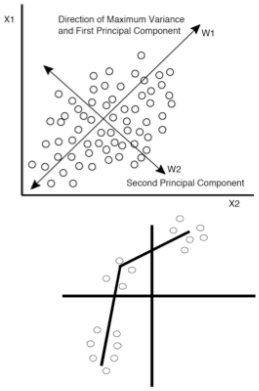
version, name, file, format, OID, title, artist, album, year, genre, comment, bit_rate, frame_rate, duration, peakAvg, peakVar, peakVar2, peakDel, peakDel2, rmsAvg, rmsVar, rmsVar2, rmsDel, rmsDel2, lp_rmsAvg, lp_rmsVar, lp_rmsVar2, lp_rmsDel, lp_rmsDel2, hp_rmsAvg, hp_rmsVar, hp_rmsVar2, hp_rmsDel, hp_rmsDel2, dyn_rangeAvg, dyn_rangeVar, dyn_rangeVar2, dyn_rangeDel, dyn_rangeDel2, stereoAvg, stereoVar, stereoVar2, stereoDel, stereoDel2, chroMaxAvg, chroMaxVar, chroMaxVar2, chroMaxDel, chroMaxDel2, chroWhtAvg, chroWhtVar, chroWhtVar2, chroWhtDel, chroWhtDel2, spectCentAvg, spectCentVar, spectCentVar2, spectCentDel, spectCentDel2, spectRollAvg, spectRollVar, spectRollVar2, spectRollDel, spectRollDel2, spectSpreadAvg, spectSpreadVar, spectSpreadVar2, spectSpreadDel, spectSpreadDel2, spectSkewAvg, spectSkewVar, spectSkewVar2, spectSkewDel, spectSkewDel2, spectKurtAvg, spectKurtVar, spectKurtVar2, spectKurtDel, spectKurtDel2, spectSlopeAvg, spectSlopeVar, spectSlopeVar2, spectSlopeDel, spectSlopeDel2, spectVarAvg, spectVarVar, spectVarVar2, spectVarDel, spectVarDel2, harmonicityAvg, harmonicityVar, harmonicityVar2, harmonicityDel, harmonicityDel2, inharmonicityAvg, inharmonicityVar, inharmonicityVar2, inharmonicityDel, inharmonicityDel2, harmonicCentAvg, harmonicCentVar, harmonicCentVar2, harmonicCentDel, harmonicCentDel2, loud, quiet, decay, held, fp_gravity, fp_bass, fp_focus, bh_lowpeakamp, bh_midpeakamp, bh_highpeakamp, bh_lowpeakBPM, bh_midpeakBPM, bh_highpeakBPM, bh_pdcentroid1, bh_pdcentroid2, bh_pdspread1, bh_pdspread2, bh_tempo, beat_peaks, beat_slope, beat_weight, beat_sum, beat_height, beat_hist_rat0, beat_hist_rat1, beat_hist_rat2, beat_hist_rat3, beat_max, bands0Avg, bands0Var, bands0Var2, bands0Del, bands0Del2, bands1Avg, bands1Var, bands1Var2, bands1Del, bands1Del2, bands2Avg, bands2Var, bands2Var2, bands2Del, bands2Del2, bands3Avg, bands3Var, bands3Var2, bands3Del, bands3Del2, drums0Avg, drums0Var, drums0Var2, drums0Del, drums0Del2, drums1Avg, drums1Var, drums1Var2, drums1Del, drums1Del2, drums2Avg, drums2Var, drums2Var2, drums2Del, drums2Del2, drums3Avg, drums3Var, drums3Var2, drums3Del, drums3Del2, drums4Avg, drums4Var, drums4Var2, drums4Del, drums4Del2, drums5Avg, drums5Var, drums5Var2, drums5Del, drums5Del2, drums6Avg, drums6Var, drums6Var2, drums6Del, drums6Del2, mfcc0Avg, mfcc0Var, mfcc0Var2, mfcc0Del, mfcc0Del2, mfcc1Avg, mfcc1Var, mfcc1Var2, mfcc1Del, mfcc1Del2, mfcc2Avg, mfcc2Var, mfcc2Var2, mfcc2Del, mfcc2Del2, mfcc3Avg, mfcc3Var, mfcc3Var2, mfcc3Del, mfcc3Del2, mfcc4Avg, mfcc4Var, mfcc4Var2, mfcc4Del, mfcc4Del2, mfcc5Avg, mfcc5Var, mfcc5Var2, mfcc5Del, mfcc5Del2, mfcc6Avg, mfcc6Var, mfcc6Var2, mfcc6Del, mfcc6Del2, mfcc7Avg, mfcc7Var, mfcc7Var2, mfcc7Del, mfcc7Del2, mfcc8Avg, mfcc8Var, mfcc8Var2, mfcc8Del, mfcc8Del2, mfcc9Avg, mfcc9Var, mfcc9Var2, mfcc9Del, mfcc9Del2, mfcc10Avg, mfcc10Var, mfcc10Var2, mfcc10Del, mfcc10Del2, mfcc11Avg, mfcc11Var, mfcc11Var2, mfcc11Del, mfcc11Del2, mfcc12Avg, mfcc12Var, mfcc12Var2, mfcc12Del, mfcc12Del2, chromaW0, chromaW1, chromaW2, chromaW3, chromaW4, chromaW5, chroma0, chroma1, chroma2, chroma3, ssdBandMean1, ssdBandMean2, ssdBandMean3, ssdBandMean4, ssdBandMean5, ssdBandMean6, ssdBandMean7, ssdBandMean8, ssdBandMean9, ssdBandMean10, ssdBandMean11, ssdBandMean12, ssdBandMean13, ssdBandMean14, ssdBandMean15, ssdBandMean16, ssdBandMean17, ssdBandMean18, ssdBandMean19, ssdBandMean20, ssdBandMean21, ssdBandMean22, ssdBandMean23, ssdBandVar1, ssdBandVar2, ssdBandVar3, ssdBandVar4, ssdBandVar5, ssdBandVar6, ssdBandVar7, ssdBandVar8, ssdBandVar9, ssdBandVar10, ssdBandVar11, ssdBandVar12, ssdBandVar13, ssdBandVar14, ssdBandVar15, ssdBandVar16, ssdBandVar17, ssdBandVar18, ssdBandVar19, ssdBandVar20, ssdBandVar21, ssdBandVar22, ssdBandVar23, ssdBandSkew1, ssdBandSkew2, ssdBandSkew3, ssdBandSkew4, ssdBandSkew5, ssdBandSkew6, ssdBandSkew7, ssdBandSkew8, ssdBandSkew9, ssdBandSkew10, ssdBandSkew11, ssdBandSkew12, ssdBandSkew13, ssdBandSkew14, ssdBandSkew15, ssdBandSkew16, ssdBandSkew17, ssdBandSkew18, ssdBandSkew19, ssdBandSkew20, ssdBandSkew21, ssdBandSkew22, ssdBandSkew23, ssdBandKurt1, ssdBandKurt2, ssdBandKurt3, ssdBandKurt4, ssdBandKurt5, ssdBandKurt6, ssdBandKurt7, ssdBandKurt8, ssdBandKurt9, ssdBandKurt10, ssdBandKurt11, ssdBandKurt12, ssdBandKurt13, ssdBandKurt14, ssdBandKurt15, ssdBandKurt16, ssdBandKurt17, ssdBandKurt18, ssdBandKurt19, ssdBandKurt20, ssdBandKurt21, ssdBandKurt22, ssdBandKurt23, ssdBandMedian1, ssdBandMedian2, ssdBandMedian3, ssdBandMedian4, ssdBandMedian5, ssdBandMedian6, ssdBandMedian7, ssdBandMedian8, ssdBandMedian9, ssdBandMedian10, ssdBandMedian11, ssdBandMedian12, ssdBandMedian13, ssdBandMedian14, ssdBandMedian15, ssdBandMedian16, ssdBandMedian17, ssdBandMedian18, ssdBandMedian19, ssdBandMedian20, ssdBandMedian21, ssdBandMedian22, ssdBandMedian23, ssdBandMin1, ssdBandMin2, ssdBandMin3, ssdBandMin4, ssdBandMin5, ssdBandMin6, ssdBandMin7, ssdBandMin8, ssdBandMin9, ssdBandMin10, ssdBandMin11, ssdBandMin12, ssdBandMin13, ssdBandMin14, ssdBandMin15, ssdBandMin16, ssdBandMin17, ssdBandMin18, ssdBandMin19, ssdBandMin20, ssdBandMin21, ssdBandMin22, ssdBandMin23, ssdBandMax1, ssdBandMax2, ssdBandMax3, ssdBandMax4, ssdBandMax5, ssdBandMax6, ssdBandMax7, ssdBandMax8, ssdBandMax9, ssdBandMax10, ssdBandMax11, ssdBandMax12, ssdBandMax13, ssdBandMax14, ssdBandMax15, ssdBandMax16, ssdBandMax17, ssdBandMax18, ssdBandMax19, ssdBandMax20, ssdBandMax21, ssdBandMax22, ssdBandMax23, Dummy

The latest distance metrics

- Using noisy labels
- Dimensionality reduction vs clustering
  - PCA
  - FLDA
  - CURE
  - SVMs
- FLDA training and clusterer app
- Train on a couple dozen well-known genres

PCA

- Easy with clear clusters or equal density
- Often necessary to repeat with different weighting metrics to find good clustering criteria
- Statistics based on covariance matrices



Experimental Results: CURE

a) BIRCH

(centroid)

b) MST METHOD

(single link)

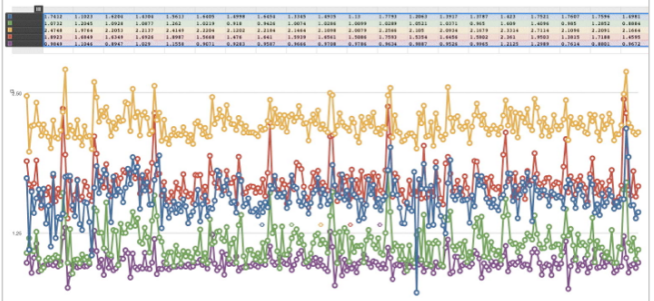
c) CURE

Picture from *CURE*, Guha, Rastogi, Shim.

Music Segmentation

- Detect onsets
- Find regular hierarchy of onsets
- Segment track into verses
- Detect intro/outro
- Detect “solo” verse or bridge
- Calculate segmentation-related features (excellent genre/style correlation)

Distance Metrics with 5 Weightings



Improving Music Information Retrieval using Segmentation

Stephen Travis Pope
UCSB Graduate Program in Media Arts and Technology, UCSB Dept. of Music

MAT Media Arts and Technology Graduate Program

CREATE

Summary

It is our goal to improve music information retrieval (MIR) software by proposing new analysis features derived from segmentation of the song content. Our process starts by segmenting a song to find the verse/chorus boundaries and identify the typical verse and solo sections. We define 15-20 new features derived from segmentation, store these in the database, and use them to prune the base feature vector, storing each feature's weighted average/variance within the typical verse.

A content-based playlist-generation system was built using the new features, with no human-supplied metadata such as playlist co-occurrence; it arguably performs better than the best "user-informed" recommenders in the field. We expect that there are many applications yet to be discovered in other domains of MIR that can use this kind of segmentation statistics and segmentation-derived higher-level song data.

Audio Analysis for MIR

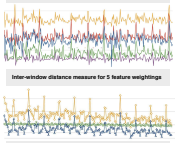
The MIR song analysis process consists of a feature extraction stage that reads audio data and performs signal analysis routines on short windows (1-50 msec), yielding a set of raw time- and frequency-domain features.

The second-pass analysis smoothes the raw data and derives higher-level features (e.g., tempo or spectral tracks), followed by pruning and reduction based on numerical/statistical analysis.

The third stage involves machine-learning or data-mining techniques such as clustering, classification or structure-learning.

Music Segmentation

Automatic music segmentation means finding the break-points between the sections of a song. For a simple case, a segmenter would collect the feature data and look for regularly spaced peaks in the weighted distance between windows. As an example, the first figure below shows five different feature weightings for 1-minute of a song. The second figure shows the auto-correlation of the distance data for several weightings; the peaks on the left are the short-time regularities (beats), and those on the right correspond to the phrases and verses. Our system uses a rather novel approach that yields particularly good results for a wide range of musical styles.



Inter-window distance measure for 5 feature weightings

Autocorrelation of several of the distance measures

Assuming a successful segmentation, we can derive an array of semantically relevant features. First, we look at the first and last regions (in-segment or not) and see if they look like fade-in/-out sections, or more like contrasting intro or "outro" sections for the song. Next, we compute the per-segment mean feature vectors to identify the "typical" and "solo" segments, i.e., both the most average and the most different. Then we compute the ratios between the verse and solo segment values for a selected set of features.

Segmentation Features

- Segmenter confidence (range 0 - 1)
- Number of segments (2 - 16 in normal music)
- Verse length (typ. 10 - 60 seconds)
- First verse start (length of intro/prelude)
- Typical/Solo start (start of the verse/solo)
- Typical/Solo index (1-16, rarely 1 or 16)
- Quiet/Loud sections (% of quiet/loud sections)
- Fade-In/Out (# secs to reach avg. amplitude)
- Solo/Verse ratios: RMS, Tempo, SpectralCentroid, SpectralVariety, DynamicRange (more possible)

Results

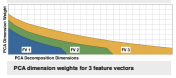
As an example of a successful segmentation, the text below is the output of a database query for some of the segmenter features of a pop/dance song (numerical features are normalized to the range 0-1).

```
| title          | segmentweight | numsegments
| I Believe In Love |     0.923772 |      0.24
| verselength | typicalstart | solostart
|     0.601119 |     0.280232 |     0.586672

| s_centroid | s_variety | s_tempo | s_dynrange
|     0.4991 |  0.001422 |  0.3360 |   0.454435
```

One measure of the quality of the analysis is the number of PCA components whose weight exceeds a given threshold. We repeat PCA with three subsets of the feature vector:

- FV1 = 26 base-features and their variances
- FV2 = FV1 + beat histogram and flux-patterns
- FV3 = FV2 + 17 segmentation-derived features



PCA Decomposition Dimensions

PCA dimension weights for 3 feature vectors

Applications

As another test, we derived PART decision lists, a data-mining technique that "discovers" weighted rules describing the data set; the new features play a role in many PART rules, e.g.,

ZeroCrossingsVar <= 0.115844 AND
fp_bass <= 0.219083 AND
LoudSections <= 0.003411 AND
NumSegments <= 0.04 AND
HPRMS <= 0.681665 --> New Age

We used the new features in a playlist generation system and compared the results to the output of systems that use human-supplied collaborative filtering metadata; we can say that the content-only solution is quite competitive with the best of them.

Playlist for Blondie, Rapture (disco/rap)

```
93:          Talking Heads --   Once In A Lifetime
94:           Roxy Music   --   The Space Between
63:            Sam Harper  --   Homeless Child
63: Alice Renee & Dois     --   It Was_A Rock This
70: August Campbell And    --   The 7-90 Song
73:        The Elementos   --   Clarinet Tentar
74:            unknown     --   Everybody Got A Long
75:  Daniel Johnston       --   I Remember Painfully
75: They Might Be Giants   --   Whistling In The Dar
80: Professor Michael DC   --   Ja
84: Bootie & The Blindfis  --   Falsewaiter Johnson
84:     The Art Of Noise   --   Kiss (Featuring Tom
```

Playlist for Glenn Gould, Bach WTC 1:2 (classical)

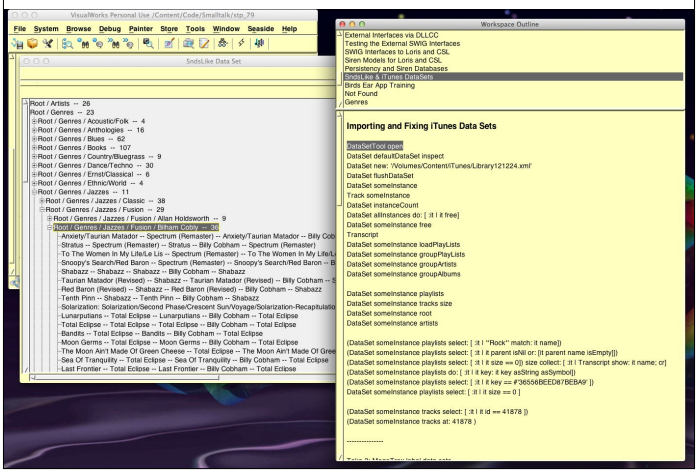
```
30:       Andrea Segovia   --   Passen Ex. 3
41:        Glenn Gould     --   Variations XXVI
68:     Sarabel: Fiesnah   --   Accompagnato
59:       Andrea Segovia   --   Passen Ex. 6
63:       Andrea Segovia   --   Prelude
66:            unknown     --   Saturday Prelude
66:       Jamie Robertson  --   Prelude Suite
66:            unknown     --   Saturday Reprise
67: Sing Creaky Fresh Si   --   O Little Town Of Bet
70:  Spencer Ma Gardens    --   Lulu Interlude
70:  Bootie & The Blindfis --   Sometimes I Feel
72:        Glenn Gould     --   Prelude in D minor
```

We have also observed the power of the new features in both CURE and Oracle database clusterers, an SVM-based classifier, and several other media content data-mining techniques.

SndsLike Development Process

- Analysis core in C++: RMS & FFT features
- Wrapper in Python
- Call-outs to Java (SSD) and Octave (FP) code
  - Really painless!
- Simple tests
  - Feature extraction
  - DB inserts
- Higher-level features
  - Rhythm, key, bass line, SSDs, etc.

Smalltalk Tools for SndsLike



Calling C from Python

```
#--- Load C API functions

  # load analysis library and its C API functions

  libanal = cdll.LoadLibrary(self.conf.analysisLib)

  # unsigned convert_file(char * in_name, char * out1_name,
  #                       char * out2_name, unsigned rate, float dur)

  self.convert_fcn = getattr(libanal, 'convert_file')

  self.convert_fcn.argtypes = [c_char_p, c_char_p, c_char_p,
        c_uint, c_float]

  self.convert_fcn.restype = c_uint
```

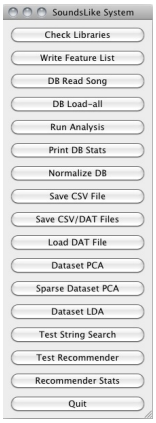

DB insert from Analysis

INSERT into FSongs (version, name, file, format, album, artist, title, genre, bit_rate, frame_rate, year, duration, rmsAvg, rmsVar, rmsDel, rmsDel2, stereoAvg, stereoVar, stereoDel, stereoDel2, spectCentAvg, spectCentVar, spectCentDel, spectCentDel2, lp_rmsAvg, lp_rmsVar, lp_rmsDel, lp_rmsDel2, spectSkewAvg, spectSkewVar, spectSkewDel, spectSkewDel2, hp_rmsAvg, hp_rmsVar, hp_rmsDel, hp_rmsDel2, chroWhtAvg, chroWhtVar, chroWhtDel, chroWhtDel2, peakAvg, peakVar, peakDel, peakDel2, spectSpreadAvg, spectSpreadVar, spectSpreadDel, spectSpreadDel2, spectKurtAvg, spectKurtVar, spectKurtDel, spectKurtDel2, chroMaxAvg, chroMaxVar, chroMaxDel, chroMaxDel2, spectVarAvg, spectVarVar, spectVarDel, spectVarDel2, spectRollAvg, spectRollVar, spectRollDel, spectRollDel2, spectSlopeAvg, spectSlopeVar, spectSlopeDel, spectSlopeDel2, fp_focus, bh_pdspread1, bh_pdspread2, fp_bass, fp_gravity, bh_midpeakamp, bh_lowpeakBPM, bh_highpeakamp, bh_pdcentroid1, bh_midpeakBPM, bh_lowpeakamp, bh_pdcentroid2, bh_highpeakBPM, bh_tempo, bands0Avg, bands0Var, bands0Del, bands0Del2, bands1Avg, bands1Var, bands1Del, bands1Del2, bands2Avg, bands2Var, bands2Del, bands2Del2, bands3Avg, bands3Var, bands3Del, bands3Del2, drums0Avg, drums0Var, drums0Del, drums0Del2, drums1Avg, drums1Var, drums1Del, drums1Del2, drums2Avg, drums2Var, drums2Del, drums2Del2, drums3Avg, drums3Var, drums3Del, drums3Del2, drums4Avg, drums4Var, drums4Del, drums4Del2, drums5Avg, drums5Var, drums5Del, drums5Del2, drums6Avg, drums6Var, drums6Del, drums6Del2, mfcc0Avg, mfcc0Var, mfcc0Del, mfcc0Del2, mfcc1Avg, mfcc1Var, mfcc1Del, mfcc1Del2, mfcc2Avg, mfcc2Var, mfcc2Del, mfcc2Del2, mfcc3Avg, mfcc3Var, mfcc3Del, mfcc3Del2, mfcc4Avg, mfcc4Var, mfcc4Del, mfcc4Del2, mfcc5Avg, mfcc5Var, mfcc5Del, mfcc5Del2, mfcc6Avg, mfcc6Var, mfcc6Del, mfcc6Del2, mfcc7Avg, mfcc7Var, mfcc7Del, mfcc7Del2, mfcc8Avg, mfcc8Var, mfcc8Del, mfcc8Del2, mfcc9Avg, mfcc9Var, mfcc9Del, mfcc9Del2, mfcc10Avg, mfcc10Var, mfcc10Del, mfcc10Del2, mfcc11Avg, mfcc11Var, mfcc11Del, mfcc11Del2, mfcc12Avg, mfcc12Var, mfcc12Del, mfcc12Del2, chromaW0, chromaW1, chromaW2, chromaW3, chromaW4, chromaW5, chromal0, chromal1, chromal2, chromal3, Normalized) — No SSD features

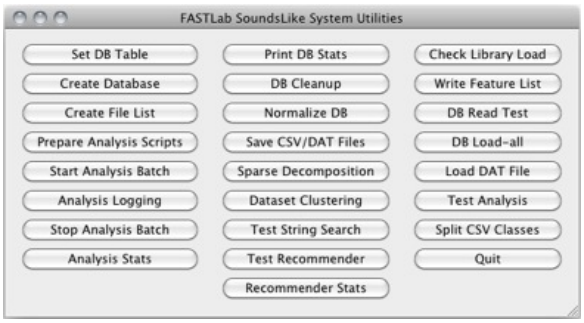
VALUES (120606, "01 - Bye Bye Blues", "FASTLab/Jazz/Bebop/Lionel Hampton/Mostly Blues/01 - Bye Bye Blues.mp3", "mp3", "Mostly Blues", "Lionel Hampton", "Bye Bye Blues", "Bebop", 192, 44100, 1939, 542, 0.0, 2.01947212219, 0.0, 0.0, 1.04739511013, 0.0106823779643, 0.000100552490039, -2.10831467484e-06, 1966.97631836, 737.2734375, 0.134124577045, -0.0296357311308, 0.925902187824, 0.0444750748575, 9.49300374486e-05, 2.60835918198e-07, 62.5583305359, 3.06155061722, 0.00632237270474, 0.000141201744555, 2.48503637314, 0.846544265747, 0.000191311017261, 3.23132080666e-05, 1.9175138732e-05, 0.508487761021, 0.0, 0.0, 0.0642576143146, 0.0304333139211, 9.53992184805e-06, 3.72650833924e-06, 0.0, 0.0, -1.25114755386e+31, 0.0, -269.560760498, 818.076049805, 0.0330554507673, 0.0235531665385, 133.833892822, 24.5230522156, 0.0108523909003, -0.000898557365872, 0.89778894186, 0.505137622356, 2.46245272137e-05, -1.0223739082e-05, 6946.84033203, 1918.30126953, 0.456291377544, -0.106259636581, 0.862270057201, 0.0490319021046, 7.62149284128e-05, -2.60426440946e-06, 0.22867, 0.331161826849, 0.294987589121, 962.769, 12.5446, 714.501403809, 75.25, 940.655517578, 0.559642314911, 100.25, 732.064941406, 0.583415329456, 151.0, 151.0, 0.00860360916704, 0.0110074682161, 5.33790580448e-06, 1.2847697235e-06, 0.0198586396873, 0.00968621950597, 1.4095280676e-06, 2.3057754106e-07, 0.00794597156346, 0.00999822001904, 1.0011528957e-06, -6.70289992399e-08, 0.00897021777928, 0.00928762741387, 7.16028978331e-07, -1.00168904282e-07, 0.006709375884 38, 0.00792932789773, 2.87757166006e-06, 3.23508402289e-07, 0.00474860239774, 0.0060529964976, 2.52516656474e-06, 1.34117385642e-06, 0.00656200340018, 0.0053910552524, 5.07121853843e-07, -1.20462630093e-07, 0.00134923018049, 0.00389251182787, 7.79723094979e-08, -9.76553309329e-08, 0.00407208828256, 0.00624277582392, 6.66937467031e-07, 1.87494421766e-07, 0.00879082549363, 0.012737242505, 6.62426828058e-07, -1.07407579719e-07, 0.0110027724877, 0.0117821972817, 1.00679585557e-06, -1.77346706209e-07, -179.230361938, 11.6381502151, -0.0174401570112, -0.00014773933799, -27.6861171722, 3.32833960388, -0.00249142223038, 9.66711086221e-05, 45.7213783264, 4.61076211929, 0.00470856018364, 0.000102287405753, -26.3424491882, 3.86742663383, -0.00193355290685, -3.23435378959e-05, 19.4510784149, 2.00444865227, 0.00170343567152, 3.27975394612e-05, -4.14126777649, 1.61075496674, -0.0001644888398, -0.00011113236 3148, -6.80255937576, 1.98178017139, -0.000444941077149, -3.22821506415e-05, 13.4546375275, 2.00520682335, 0.00117843493354, 7.6199270552e-05, -12.2699079514, 1.78280282021, -0.000979631091468, 6.57258278807e-05, 9.10339832306, 1.80766713619, 0.00100279122125, 6.51373848086e-05, -3.96917152405, 1.85294151306, -0.000161508185556, -6.8761924922e-05, -3.31718492508, 1.4685356617, -0.000212517756154, 1.02561052699e-05, 7.42214632034, 1.53596329689, 0.000954209419433, 6.28527632216e-05, 0.0331983529031, 0.0286151412874, 0.0207273643464, 0.0147993322462, 0.0118453074247, 0.0116100925952, 1.0, 4.0, 6.0, 8.0, 0);

Testing/Demo GUIs

- Test GUI button panel
  - In Python & Qt
- Various data plots
  - Several tools: gnuplot, XL, etc.
- MySQL tests
- Demo "Player" GUI
  - C++/JUCE



OPs/Test GUI



SndsLike Player

Dist	Title	Artist	Album	Genre	Duration
0	This Flight Tonight	Joni Mitchell	Blue	Folk	2:53
0	River	Joni Mitchell	Blue	Folk	4:05
0	A Case Of You	Joni Mitchell	Blue	Folk	4:23
0	The Last Time I Saw Richard	Joni Mitchell	Blue	Folk	4:17
0	Court And Spark	Joni Mitchell	Court And Spark	Folk	2:46
0	Help Me	Joni Mitchell	Court And Spark	Folk	3:22
0	Peoples Parties	Joni Mitchell	Court And Spark	Folk	2:15
0	Same Situation	Joni Mitchell	Court And Spark	Folk	2:57
0	Car On A Hill	Joni Mitchell	Court And Spark	Folk	3:02
0	Down To You	Joni Mitchell	Court And Spark	Folk	5:39
0	Just Like This Train	Joni Mitchell	Court And Spark	Folk	4:25
0	Raised On Robbery	Joni Mitchell	Court And Spark	Folk	3:07

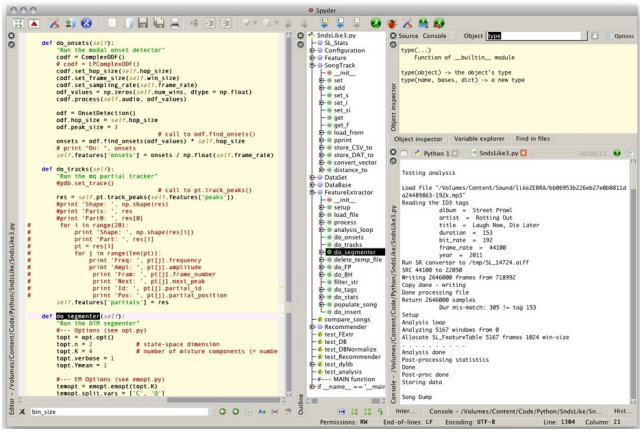
Data Sets

- FASTLab - 14 kSongs, very diverse, “high-quality,” well-encoded
- LikeZebra - 250 kSongs pop/rock
- MegaTrax - 160 kSongs + stems
- AudioNet - 200 kSongs + stems
- Others (not public)

Marketing

- Production music houses "still don't get it”
  - “Customers aren’t asking for this…”
- Performance Problems
  - Many versions of the same track with different instrumentations - “stems”
  - Same track with/without vocals
  - Cover songs

Code Tour in Spyder



Lessons Learned

- I still want it!
  - …so please make me one that gets accepted by the on-line services…
- They (production music houses, record labels, Gracenote, Apple, Adobe, …) still don’t get it.

- Q & A
- Thank you!