upf. Universitat Pompeu Fabra *Barcelona*

MTG
Music Technology
Group

Why Reproducibility Matters?
A Personal Experience

Sertan Şentürk

PhD Candidate – CompMusic Project

Music Technology Group, Universitat Pompeu Fabra

sertan.senturk@upf.edu

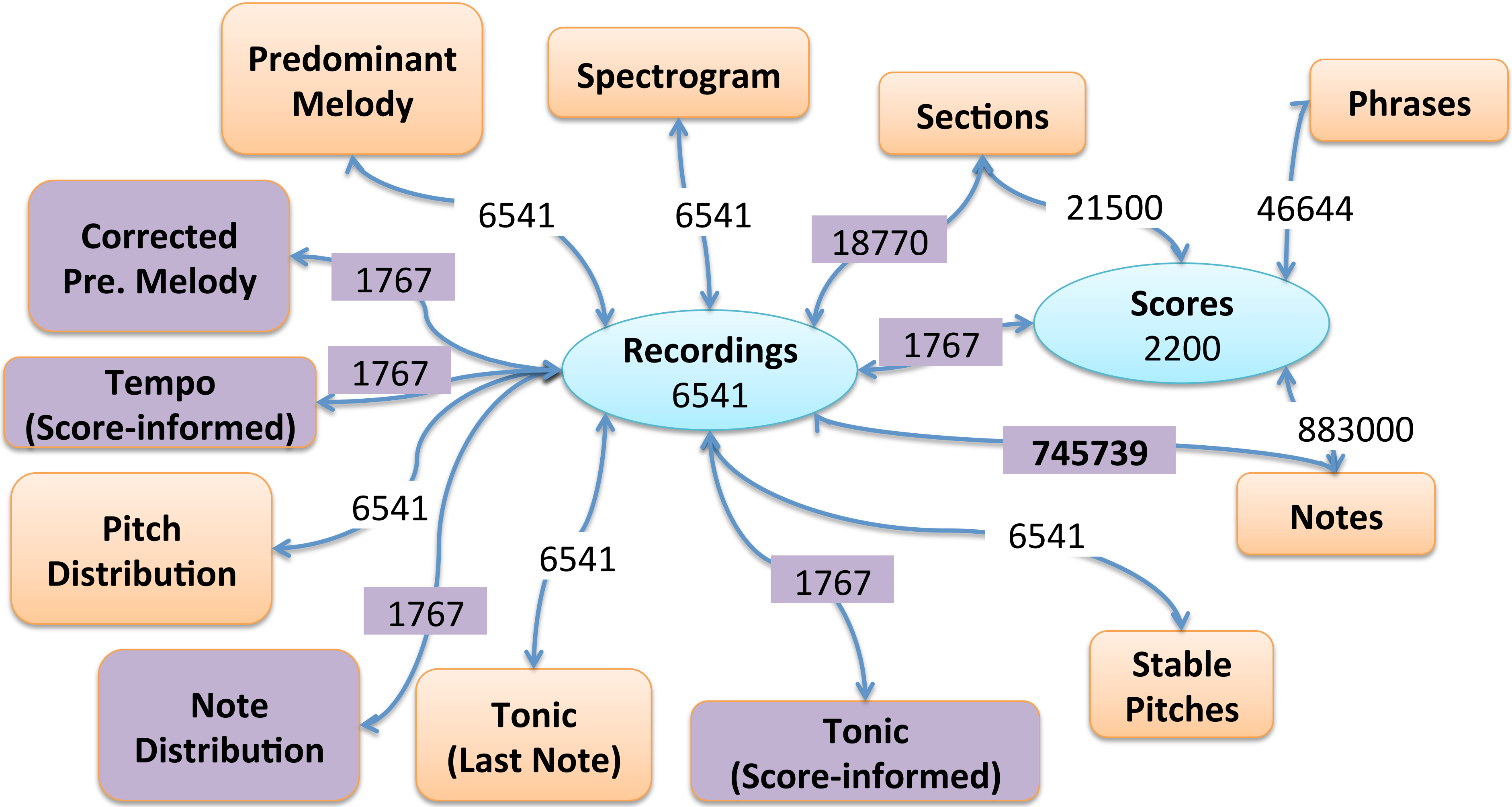
ETIC PhD Research Seminar
17 January 2017
MdM Strategic Program – Universitat Pompeu Fabra

Şentürk, S. (2016). **Computational Analysis of Audio Recordings and Music Scores for the Description and Discovery of Ottoman-Turkish Makam Music.** PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain.

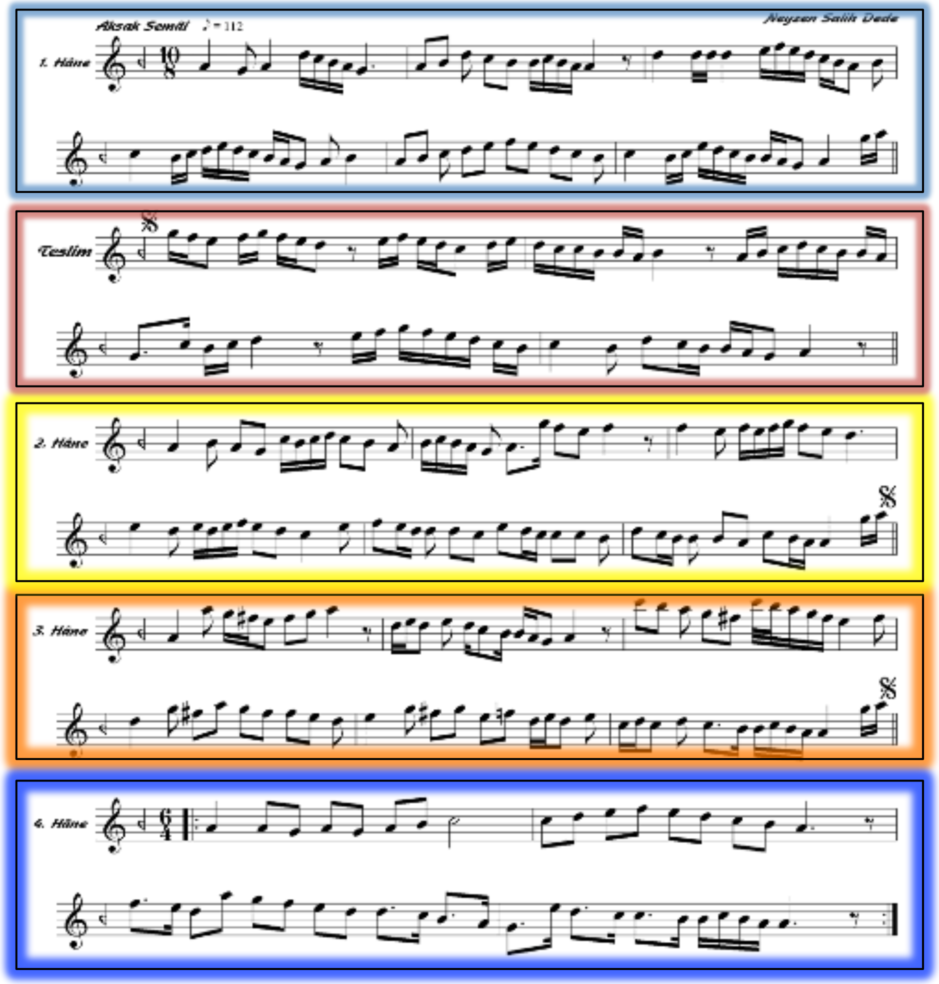
I’ll be defending on February 22nd

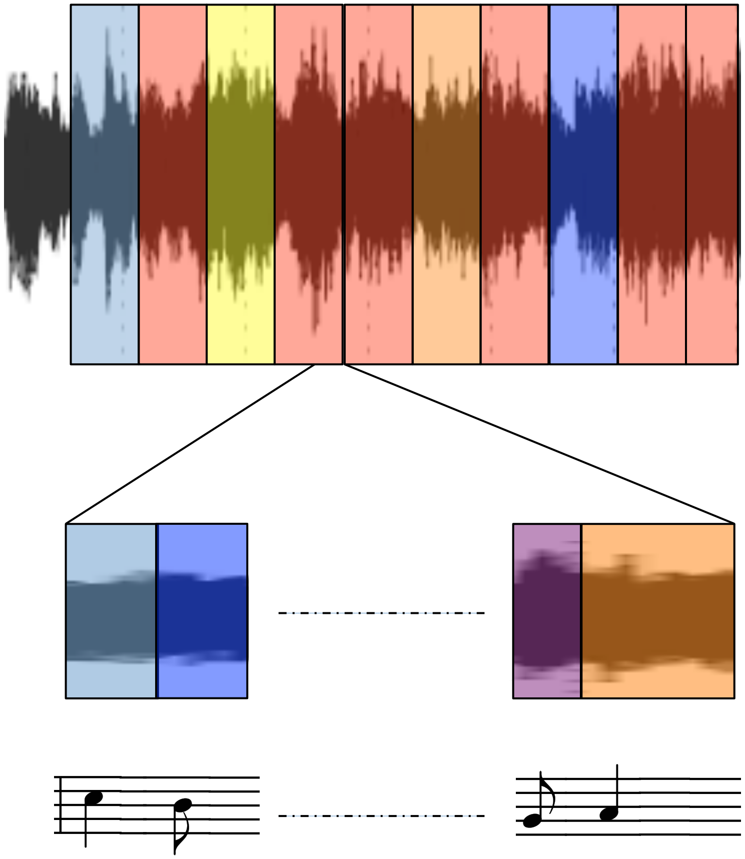
Everybody’s welcome ^_^

Computational Analysis



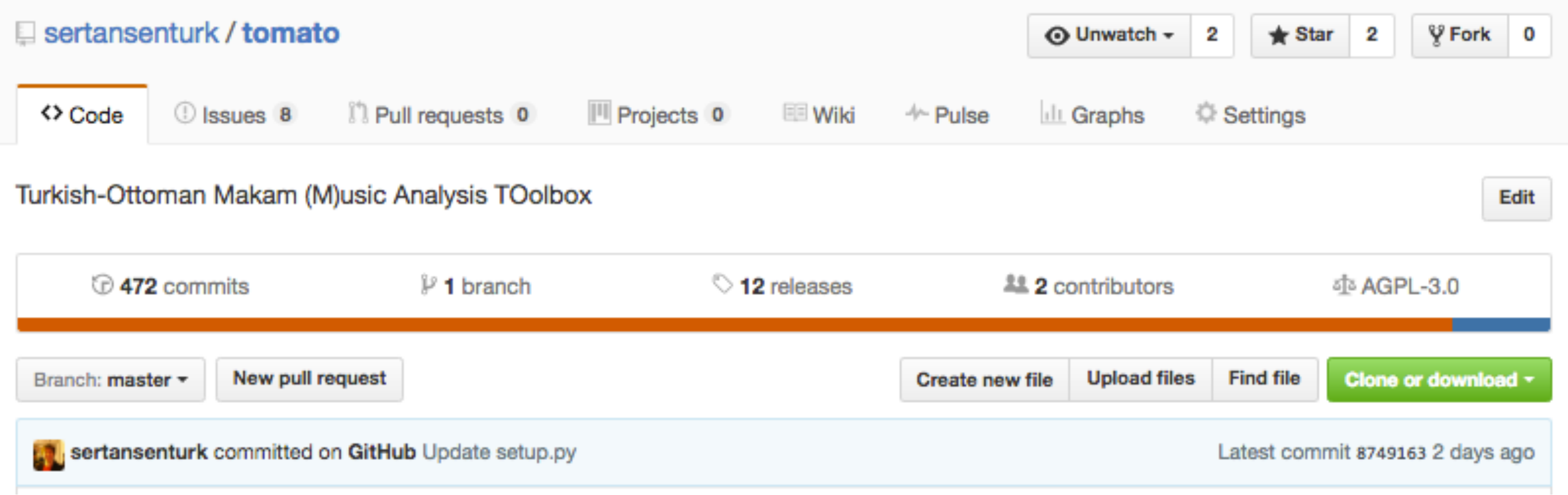
Audio-Score Alignment





Code & Results

https://github.com/sertansenturk/tomato



Turkish-Ottoman Makam (M)usic Analysis TOolbox

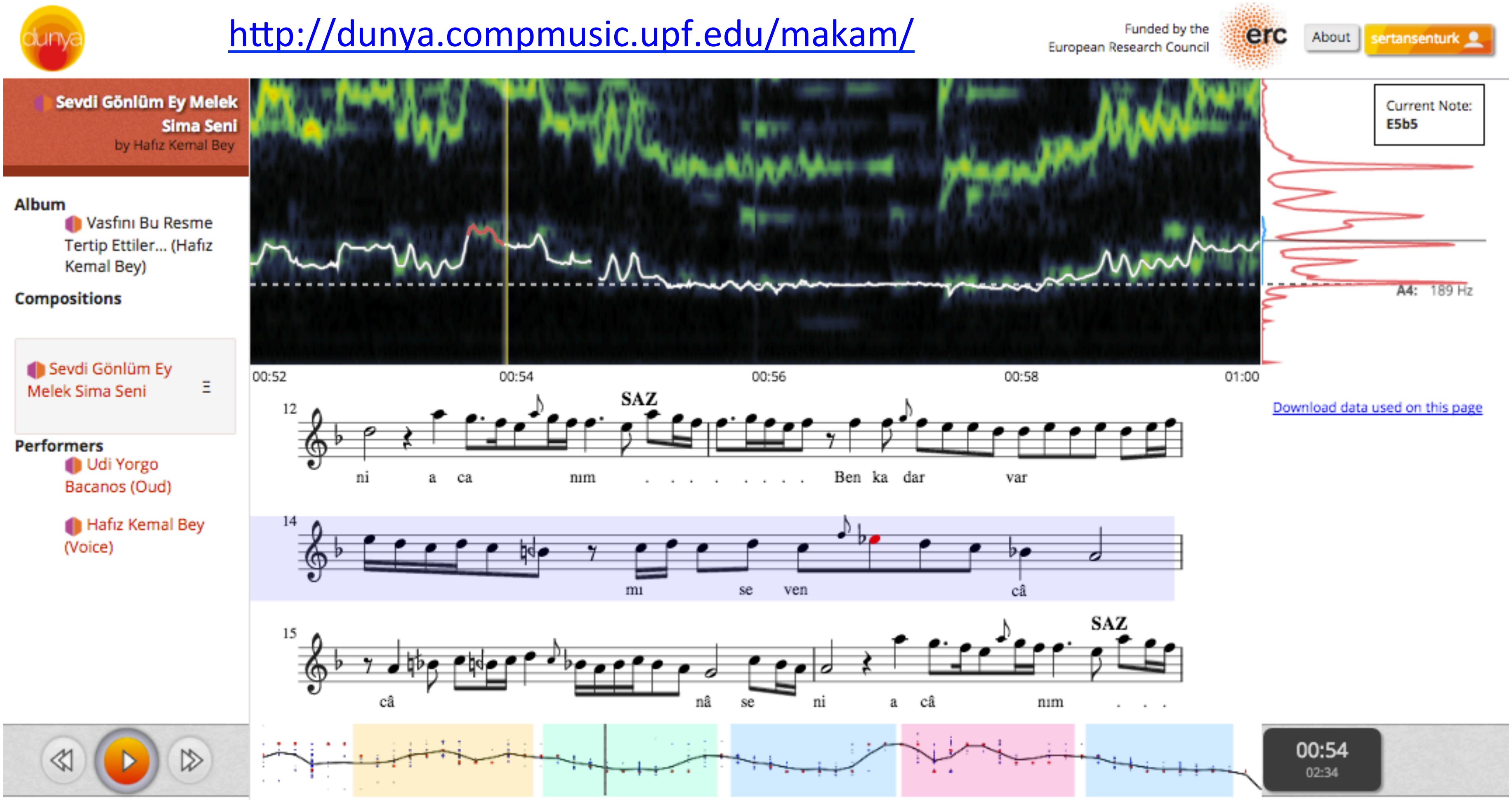
472 commits · 1 branch · 12 releases · 2 contributors · AGPL-3.0

sertansenturk committed on GitHub Update setup.py — Latest commit 8749163 2 days ago

- Analyzed a corpus consisting 2200 music scores and more than 6600 audio recordings
  - 85 hours of time-aligned audio data

Music Discovery

http://dunya.compmusic.upf.edu/makam/



Funded by the European Research Council

About

sertansenturk

Sevdi Gönlüm Ey Melek Sima Seni

by Hafız Kemal Bey

Album

Vasfını Bu Resme Tertip Ettiler... (Hafız Kemal Bey)

Compositions

Sevdi Gönlüm Ey Melek Sima Seni

Performers

Udi Yorgo Bacanos (Oud)

Hafız Kemal Bey (Voice)

Current Note: **E5b5**

Download data used on this page

Deliverables

- The datasets, code and results presented in the thesis are open.

  http://compmusic.upf.edu/senturk2016thesis

- Some of the material has already been used by other people and/or adapted to other music cultures.

Openness -> Reproducible?

Does openness really mean reproducibility?

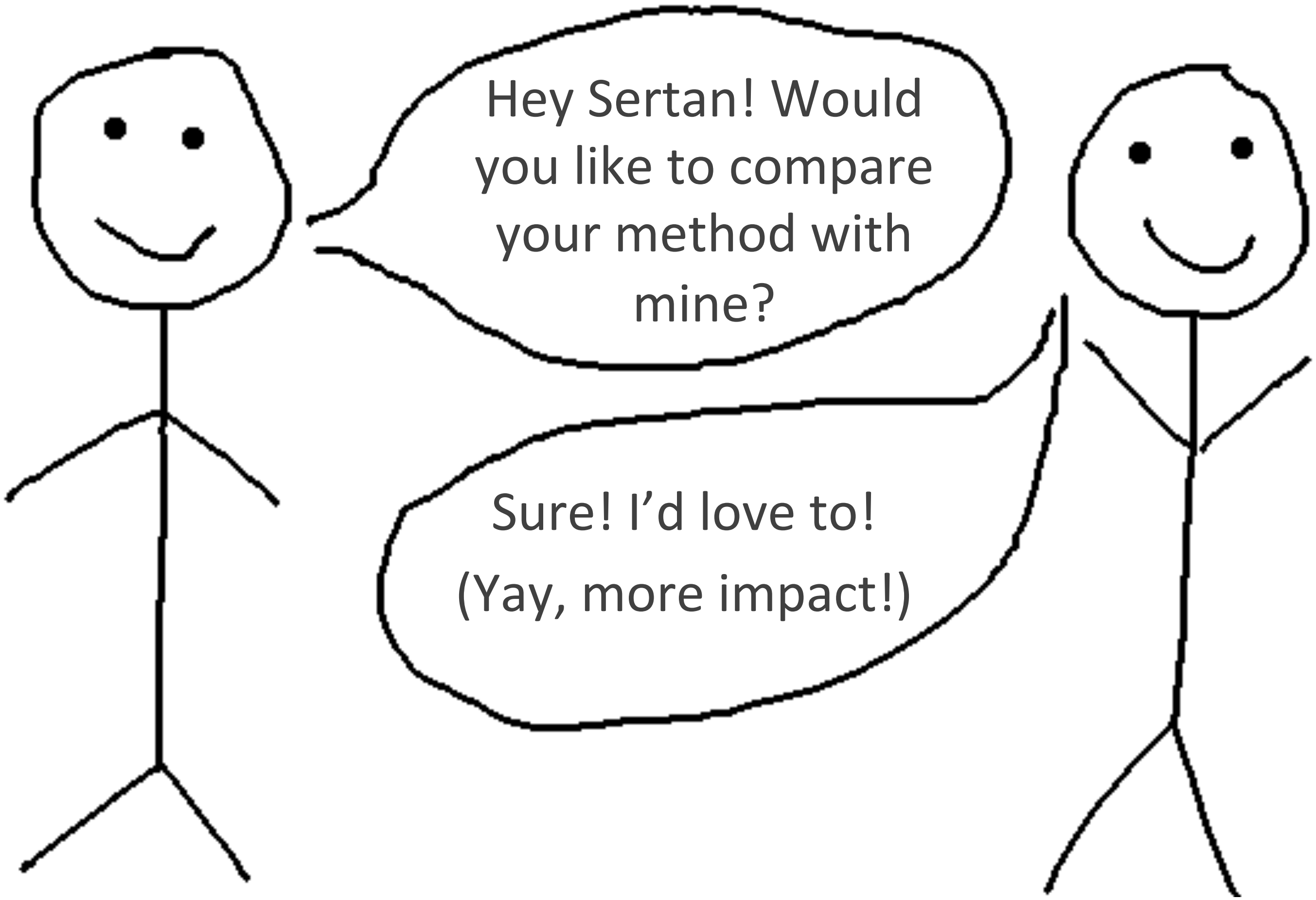
Not really ☹

*at least, not the stuff I’ve done in the start

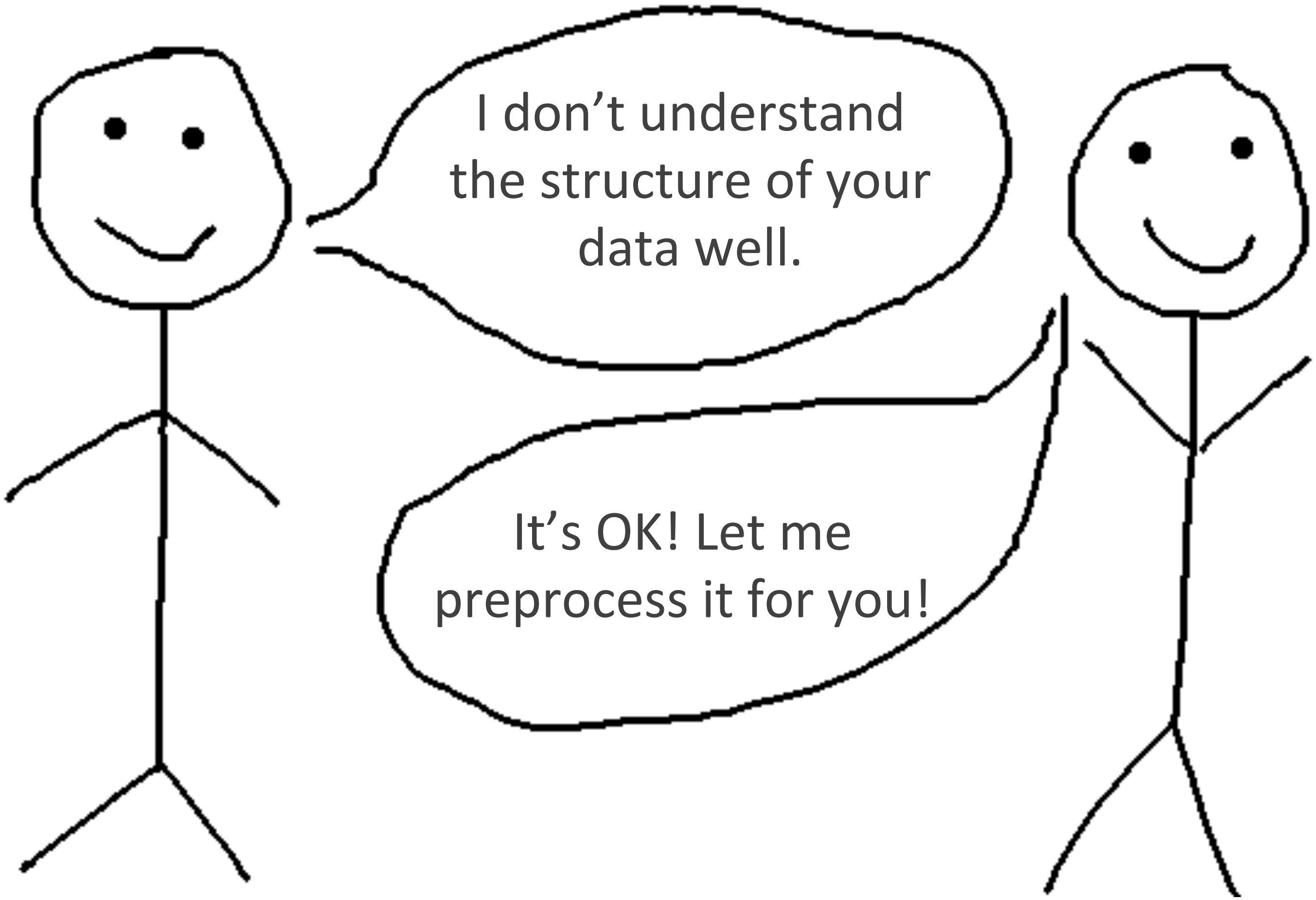
How I learned the hard way

- The story starts two years ago…
- A researcher asked me to help him to compare my previous audio-score alignment method with his new approach
- I thought “Everything is already open and online, how hard could it be?”

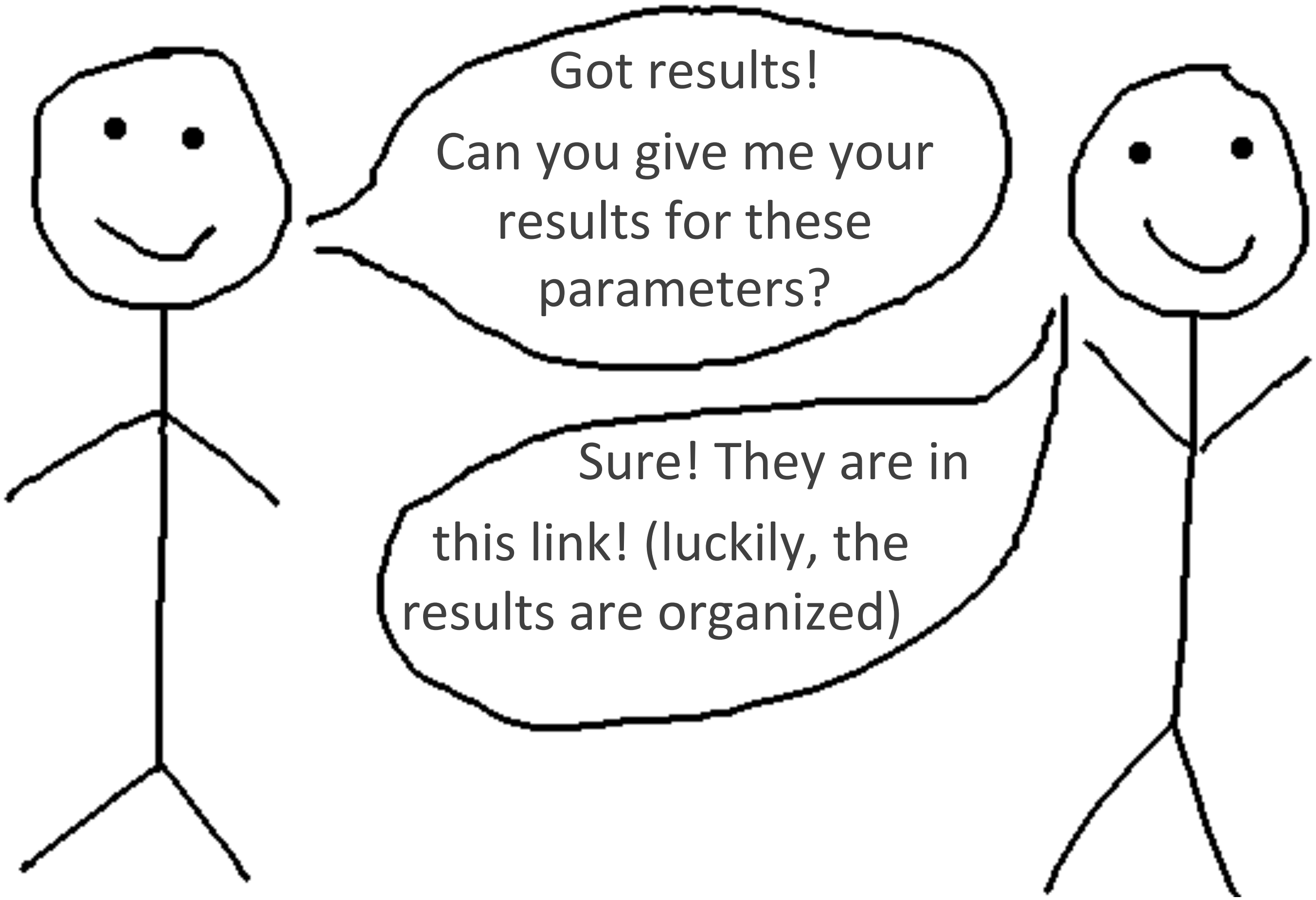
How I learned the hard way – Day 1


Hey Sertan! Would you like to compare your method with mine?
Sure! I'd love to!
(Yay, more impact!)

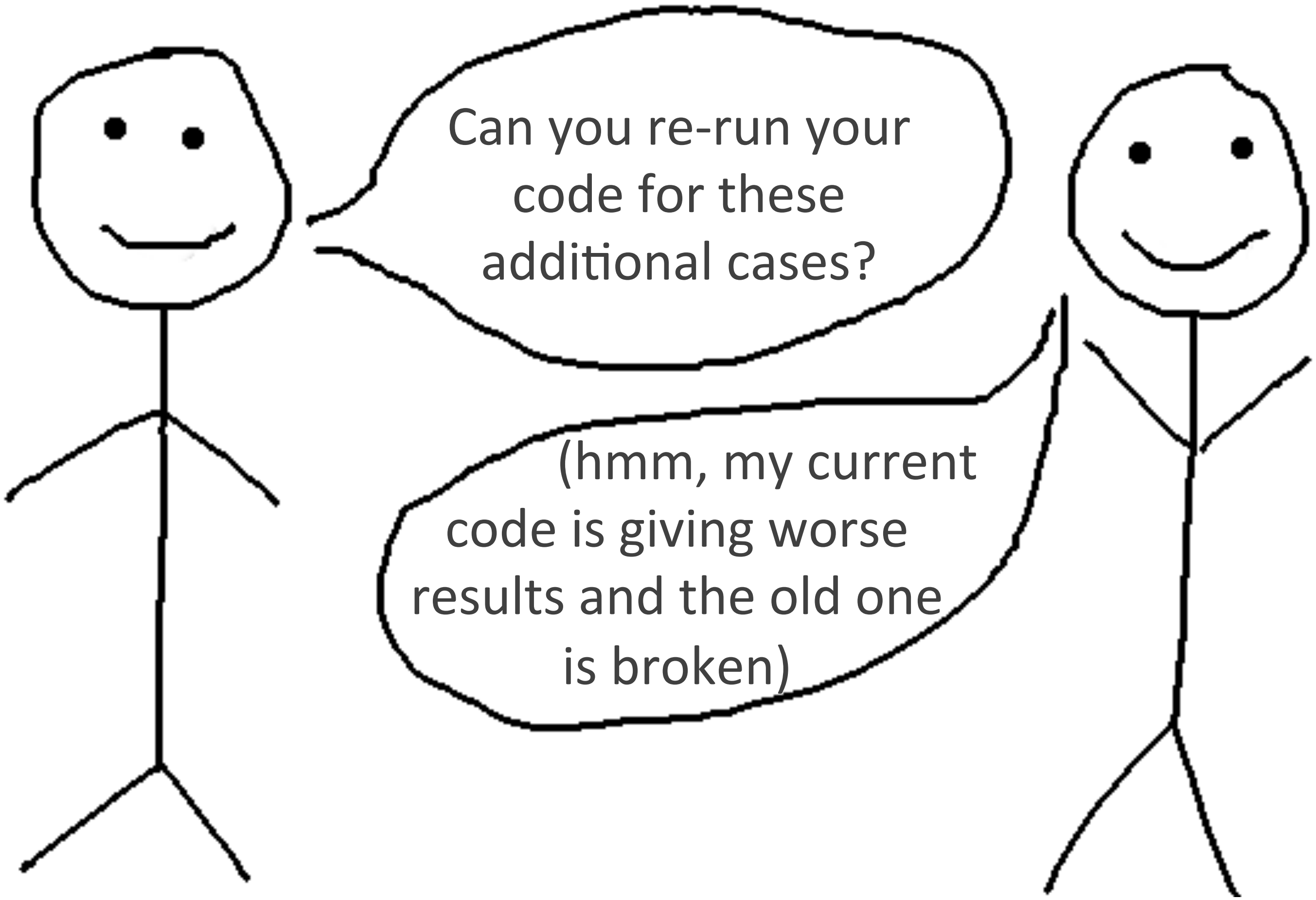
How I learned the hard way – Day 4


I don’t understand the structure of your data well.
It’s OK! Let me preprocess it for you!

How I learned the hard way – Day 7

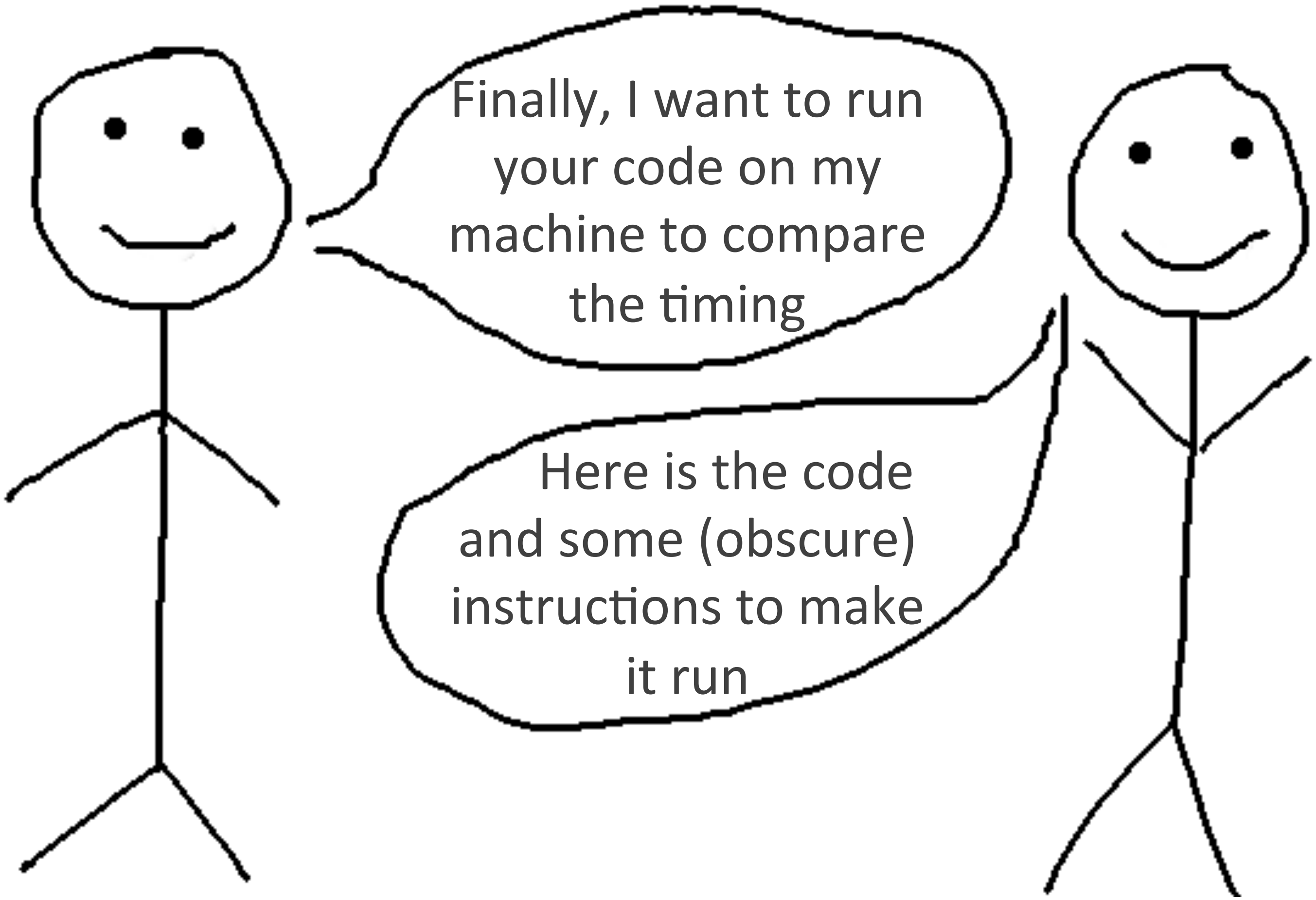

Got results!
Can you give me your results for these parameters?
Sure! They are in this link! (luckily, the results are organized)

How I learned the hard way – Day 9

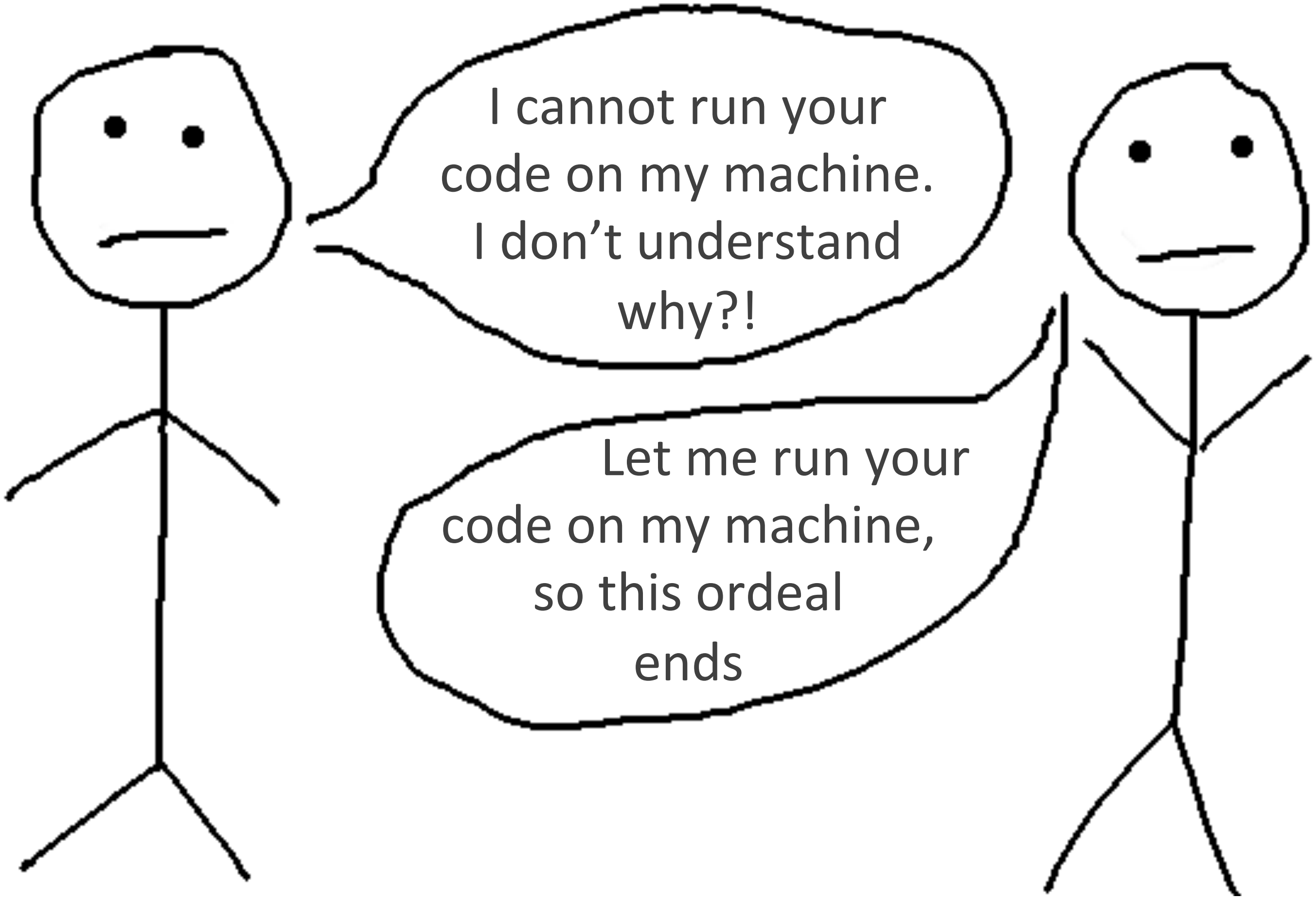

Can you re-run your code for these additional cases?
(hmm, my current code is giving worse results and the old one is broken)

How I learned the hard way – Day 15

* After (magically) fixing the old code and providing the additional results


Finally, I want to run your code on my machine to compare the timing
Here is the code and some (obscure) instructions to make it run

How I learned the hard way – Day 18


I cannot run your code on my machine. I don't understand why?!
Let me run your code on my machine, so this ordeal ends

Lessons learned

- It took us 3 weeks to reproduce my results.
  - I had spent 2 days to run the code and report the results before.
  - I could have worked on something better!
  - It's not fun to redo all that work again!
  - It's not fun (or sometimes possible) either for others to reproduce your work from scratch!
- The first person to benefit by making your work reproducible is you!

Aftermath

- Afterwards, I tried to make my work as easy to reproduce as possible
- How?
  - Always version the data, code and experiments
  - Putting effort to write more readable, modular and distributable code
  - Documenting everything clearly
  - Properly publishing all research material
- The more you put your effort, the better you will get!
  - The quality of your research output will also get better!

Disclaimers

- Open research implied
- Some suggestions are not directly related to reproducibility but will impact your work in general
- The implications are not restricted to the specific examples
  - Arguments could apply to code, data, results, publications etc.
- Inspired from the tools I use
  - Data (tsv, json …)
  - Code (Python, MATLAB, …)
  - Publications (LaTeX)
- Ideal cases
  - not always feasible/applicable

Paper Reproducibility Example

MORTY: A Toolbox for Mode Recognition and Tonic Identification

Altuğ Karakurt
The Ohio State University
Columbus, OH 43210
karakurt.1@osu.edu

Sertan Şentürk
Universitat Pompeu Fabra
Roc Boronat, 138
Barcelona, Spain 08018
sertan.senturk@upf.edu

Xavier Serra
Universitat Pompeu Fabra
Roc Boronat, 138
Barcelona, Spain 08018
xavier.serra@upf.edu

ABSTRACT

In the general sense, mode defines the melodic framework and tonic acts as the reference tuning pitch for the melody in the performances of many music cultures. The mode and tonic information of the audio recordings is essential for many music information retrieval tasks such as automatic transcription, tuning analysis and music similarity. In this paper we present MORTY, an open source toolbox for mode recognition and tonic identification. The toolbox implements generalized variants of two state-of-the-art methods based on pitch distribution analysis. The algorithms are designed in a generic manner such that they can be easily optimized according to the culture-specific aspects of the studied music tradition. We test the generalized methodology systematically on the largest mode recognition dataset curated for Ottoman-Turkish makam music so far, which is composed of 1000 recordings in 50 modes. We obtained 95.8%, 71.8% and 63.6% accuracy in tonic identification, mode recognition and joint mode and tonic estimation tasks, respectively. We additionally present recent experiments on Carnatic and Hindustani music in comparison with several methodologies recently proposed for raga/raag recognition. We prioritized the reproducibility of our work and provide all of our data, code and results publicly. Hence we hope that our toolbox would be used as a benchmark for future methodologies proposed for mode recognition and tonic identification, especially for music traditions in which these computational tasks have not been addressed yet.

Keywords

Mode recognition; Tonic Identification; Toolbox; Ottoman-Turkish makam music; Carnatic Music; Hindustani Music; Pitch Class Distribution; *k*-nearest neighbors classification; Open Source Software; Reproducibility

1. INTRODUCTION

In many music cultures, the melodies adhere to a particular melodic framework, which specifies the melodic characteristics of the music. While the function and the understanding of these frameworks are distinct from a culture-specific perspective, in a broader sense they may be considered as the "modes" of the studied music culture. Some of the music traditions that can be considered as "modal" are Indian art musics, the makam traditions and medieval church chants [20]. Mode recognition is an important complementary task in computational musicology, music discovery, music similarity and recommendation.

Tonic is another important musical concept. It acts as the reference frequency for the melodic progression in a performance. In many music cultures there is no standard reference tuning frequency, which makes it crucial to identify the tonic frequency to study melodic interactions. Estimating the tonic of a recording is the first step for various computational tasks such as tuning analysis [7], automatic transcription [4] and melodic motif discovery [16].

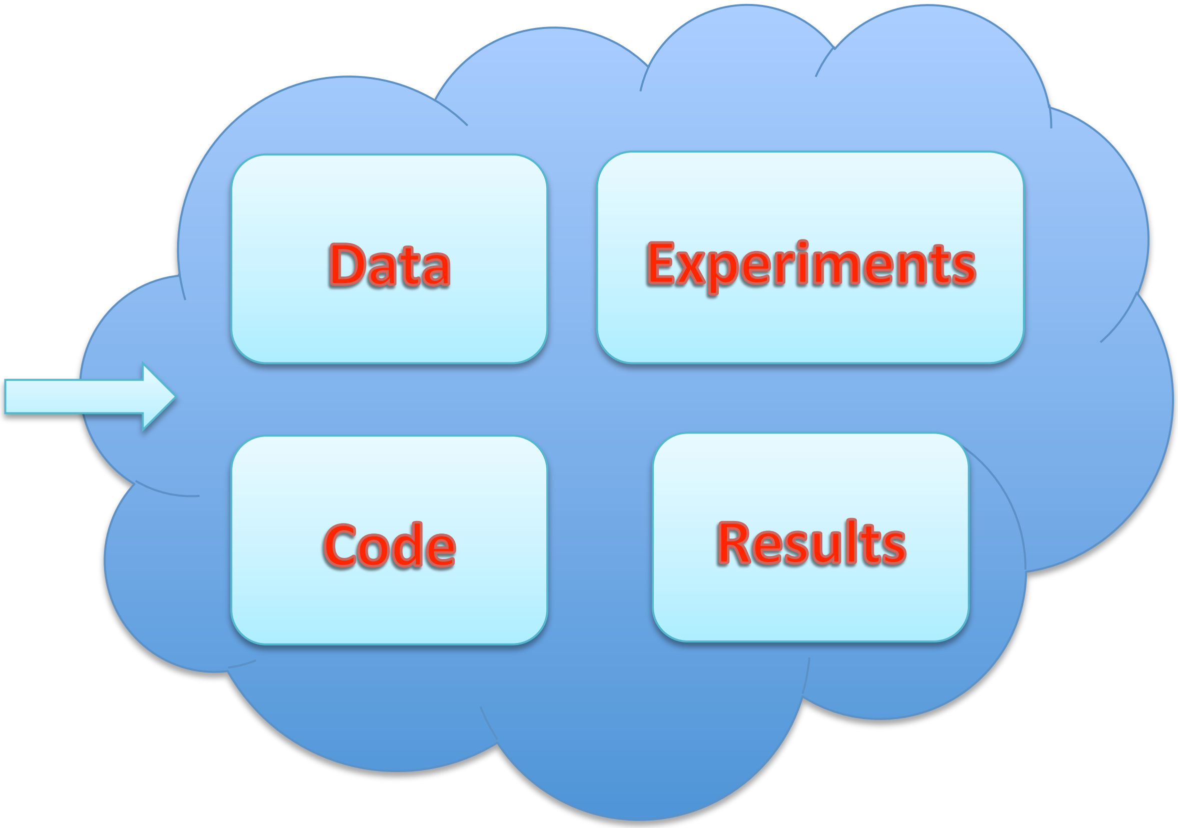
There has been a extensive interest on mode recognition in the last decade [17]. Most of these work focus on culture-specific approaches for music traditions like Ottoman-Turkish Makam music (OTMM) [13], Carnatic music [11, 12, 16], Hindustani music [9, 10, 15] and Dastgah music [1]. A considerable portion of these studies are based on comparing pitch distributions [9, 10, 11, 12, 13], which are shown to be reliable in the mode recognition task. There also exists recent approaches that are based on characteristic melodic motif mining using network analysis [15, 16], aggregating note models using automatic transcription [18] or audio-score alignment [23] and classification using neural networks [24, 26], all of which are designed specific to the studied music culture and are not generalizable to other music cultures without considerable effort. Similarly, several studies on tonic identification use pitch distribution based methods [6, 10]. More recently there has been an interest in culture specific methods for this task [2, 14, 22] that make use of heuristics and the musical characteristics of the studied tradition.

In these studies, the features extracted from the data,[1] source code and the experimental results are not usually shared. We consider the unavailability of public tools, datasets and reproducible experimentation as major obstacles for computational music information research, especially if the relevant tasks have not been applied to studied music traditions earlier.

We present MORTY (**MO**de **R**ecognition and **T**onic **Y**dentification Toolbox), an open source toolbox written in *Python*

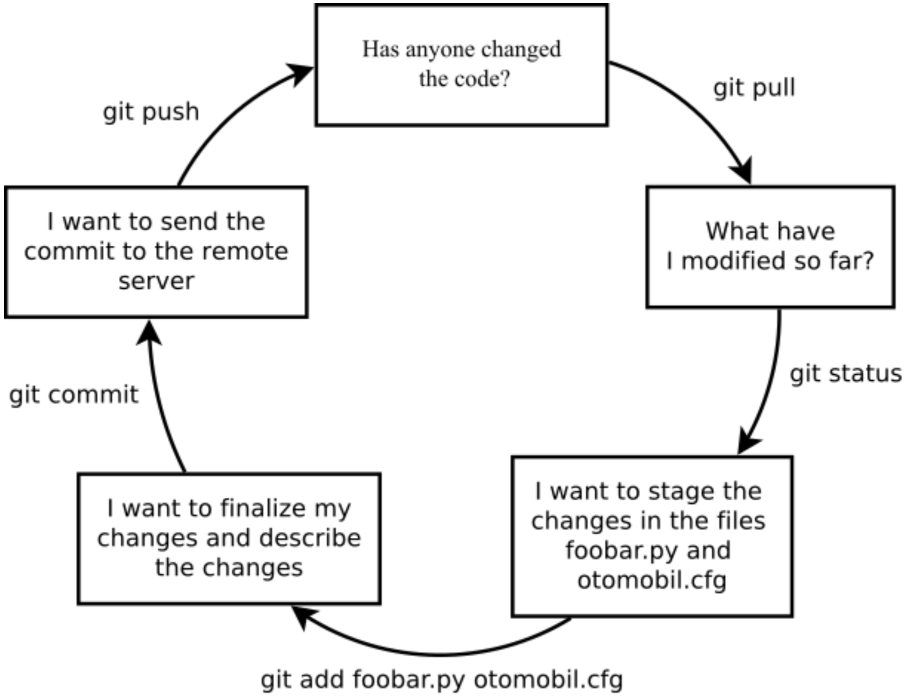
3rd International Digital Libraries for Musicology workshop (DLfM 2016)
August 12 2016, New York, USA
© 2016 ACM. ISBN 978-1-4503-4751-8/16/08…$15.00
DOI: http://dx.doi.org/10.1145/2970044.2970054

[1] Excluding the commercial audio recordings, which cannot be generally made public due to copyright laws.



Version Control

- Use git!
  - Simple
  - Reliable
  - Saves a lot of time
- Store online
  - Github (most popular)
  - Bitbucket
  - …
- No need to worry where stuff is (e.g. if your laptop gets broken)


Has anyone changed the code?

git pull

What have I modified so far?

git status

I want to stage the changes in the files foobar.py and otomobil.cfg

git add foobar.py otomobil.cfg

I want to finalize my changes and describe the changes

git commit

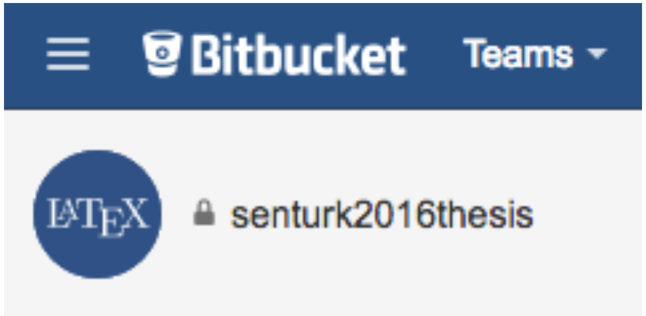
I want to send the commit to the remote server

git push

Version Control

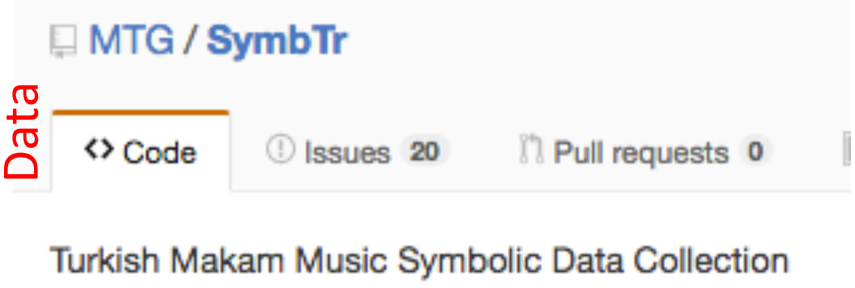
- Use git for everything!*

Publications



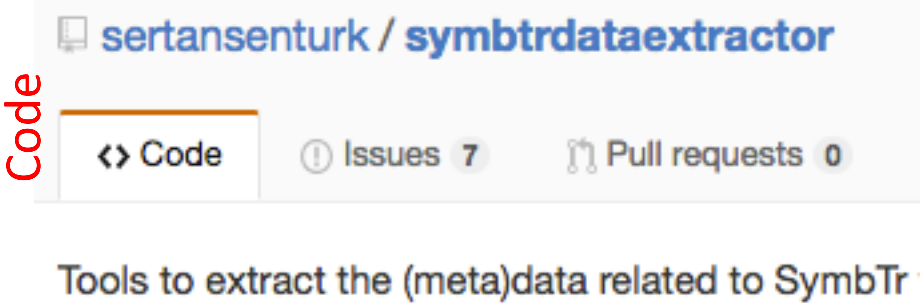
senturk2016thesis

Data


MTG / SymbTr

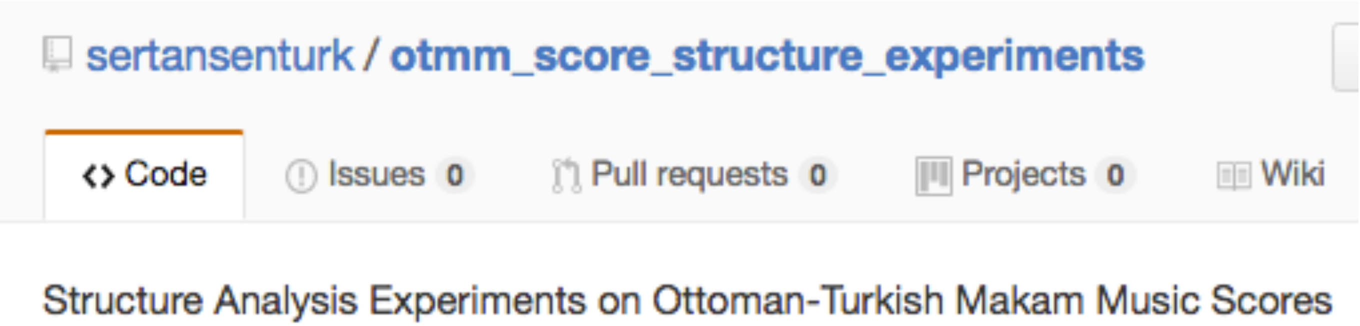
Turkish Makam Music Symbolic Data Collection

Code


sertansenturk / symbtrdataextractor

Tools to extract the (meta)data related to SymbTr

Experiments

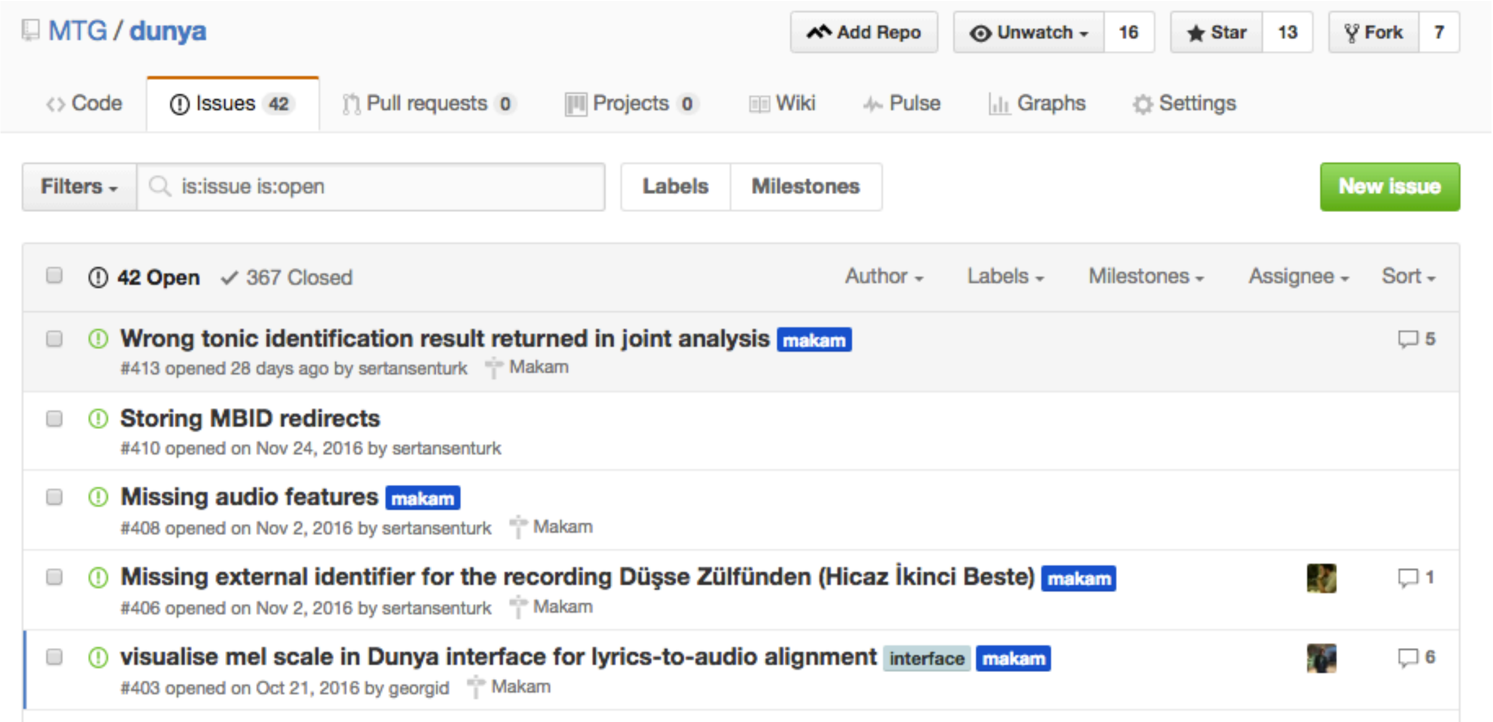

sertansenturk / otmm_score_structure_experiments

Structure Analysis Experiments on Ottoman-Turkish Makam Music Scores

* Unless, the material is big (> 1 GB)

Open Issues!

- Problems should also be documented (**not forgotten!**) and organized!
- You can discuss, track ideas, improvements etc.


MTG / **dunya**

□ 42 Open ✓ 367 Closed

□ **Wrong tonic identification result returned in joint analysis** makam — 5
#413 opened 28 days ago by sertansenturk — Makam

□ **Storing MBID redirects**
#410 opened on Nov 24, 2016 by sertansenturk

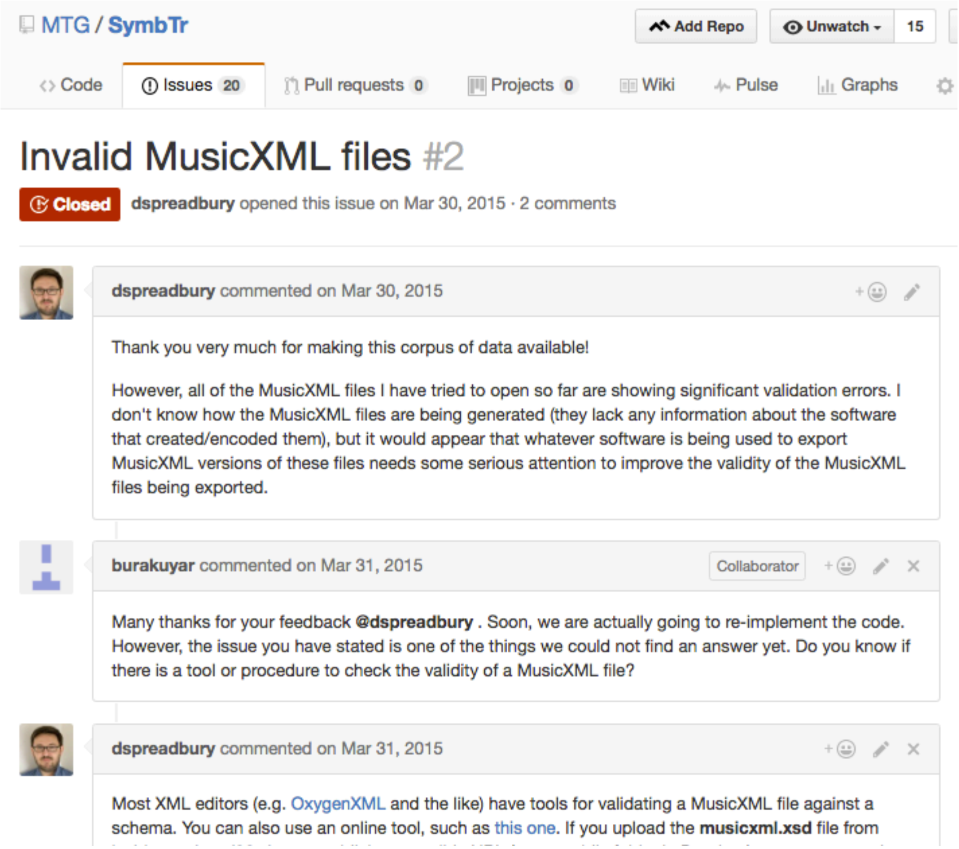
□ **Missing audio features** makam
#408 opened on Nov 2, 2016 by sertansenturk — Makam

□ **Missing external identifier for the recording Düşse Zülfünden (Hicaz İkinci Beste)** makam — 1
#406 opened on Nov 2, 2016 by sertansenturk — Makam

□ **visualise mel scale in Dunya interface for lyrics-to-audio alignment** interface makam — 6
#403 opened on Oct 21, 2016 by georgid — Makam

Active Usage of Issue Tracking Brings Contributions!

- Encourages others to get engaged in your work!



Invalid MusicXML files #2

dspreadbury opened this issue on Mar 30, 2015 · 2 comments

dspreadbury commented on Mar 30, 2015

Thank you very much for making this corpus of data available!

However, all of the MusicXML files I have tried to open so far are showing significant validation errors. I don't know how the MusicXML files are being generated (they lack any information about the software that created/encoded them), but it would appear that whatever software is being used to export MusicXML versions of these files needs some serious attention to improve the validity of the MusicXML files being exported.

burakuyar commented on Mar 31, 2015

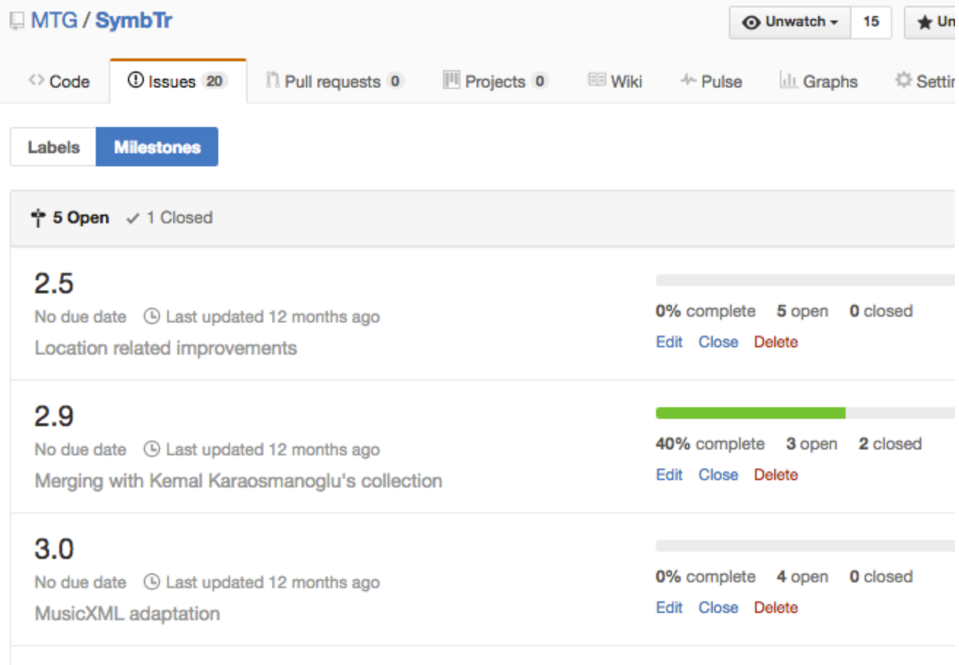
Many thanks for your feedback **@dspreadbury** . Soon, we are actually going to re-implement the code. However, the issue you have stated is one of the things we could not find an answer yet. Do you know if there is a tool or procedure to check the validity of a MusicXML file?

dspreadbury commented on Mar 31, 2015

Most XML editors (e.g. OxygenXML and the like) have tools for validating a MusicXML file against a schema. You can also use an online tool, such as this one. If you upload the **musicxml.xsd** file from

Create Milestones

- If you have to develop around a “concept” (e.g. an extension of your method to a new task), make it a milestone!
  - And subdivide into many issues
- You can track progress and be more focused



5 Open ✓ 1 Closed

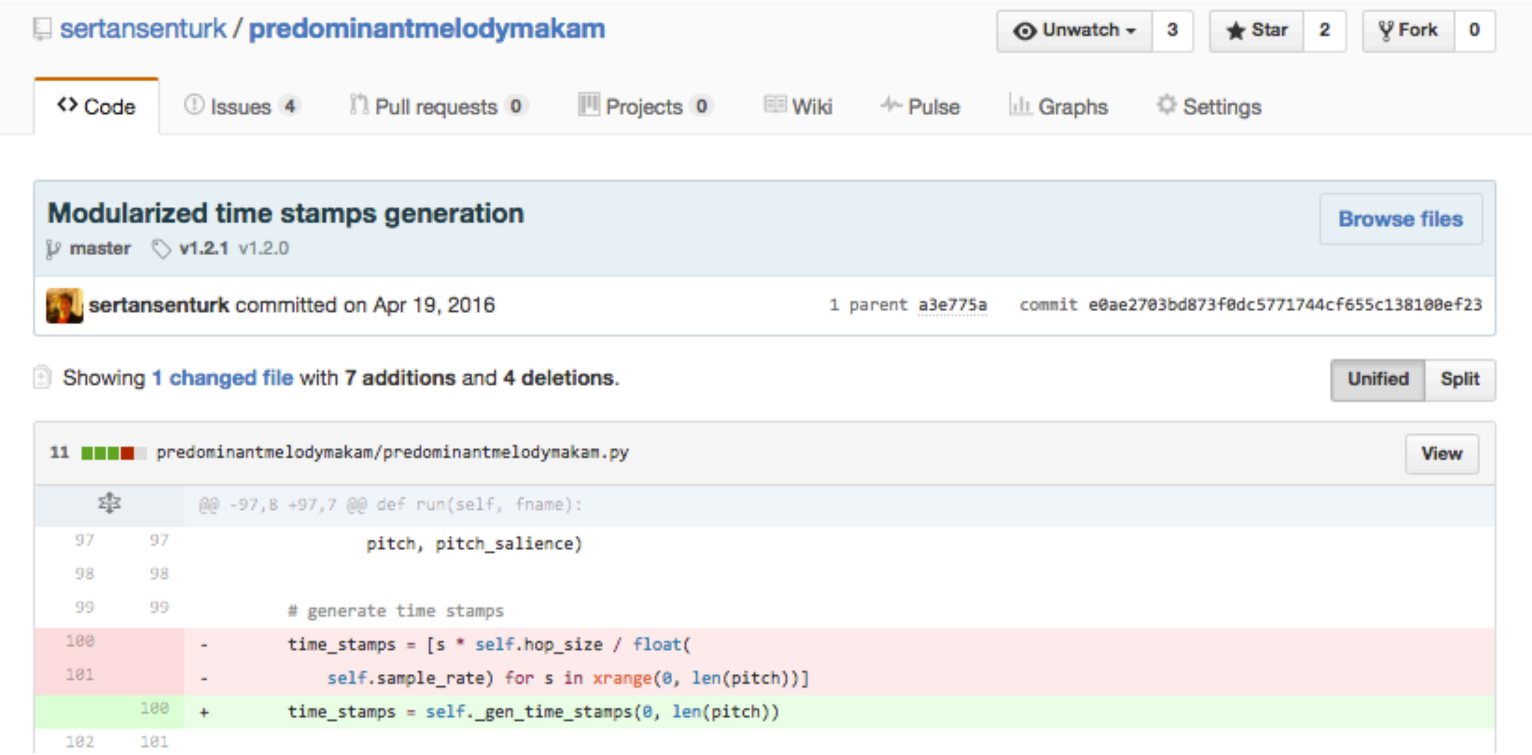
2.5
No due date Last updated 12 months ago
Location related improvements
0% complete 5 open 0 closed
Edit Close Delete

2.9
No due date Last updated 12 months ago
Merging with Kemal Karaosmanoglu's collection
40% complete 3 open 2 closed
Edit Close Delete

3.0
No due date Last updated 12 months ago
MusicXML adaptation
0% complete 4 open 0 closed
Edit Close Delete

How to commit

- Divide each problem into smaller steps
- Always make small, incremental changes/additions/fixes



Modularized time stamps generation

master v1.2.1 v1.2.0

sertansenturk committed on Apr 19, 2016

1 parent a3e775a commit e0ae2703bd873f0dc5771744cf655c138100ef23

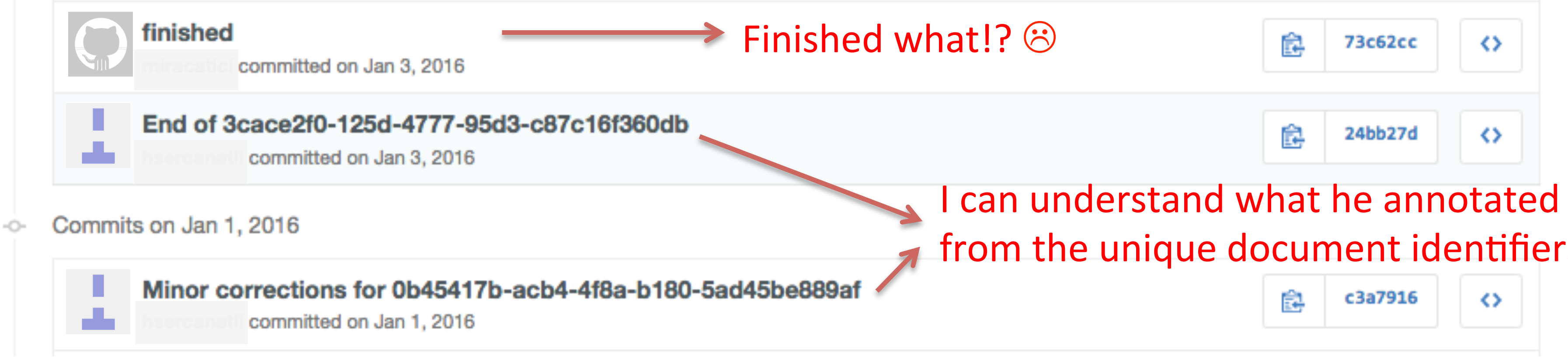
Showing 1 changed file with 7 additions and 4 deletions.

11 predominantmelodymakam/predominantmelodymakam.py

```
@@ -97,8 +97,7 @@ def run(self, fname):
                      pitch, pitch_salience)

          # generate time stamps
-         time_stamps = [s * self.hop_size / float(
-             self.sample_rate) for s in xrange(0, len(pitch))]
+         time_stamps = self._gen_time_stamps(0, len(pitch))
```

Describe your commits

- Data annotation example



finished
[redacted] committed on Jan 3, 2016
73c62cc

End of 3cace2f0-125d-4777-95d3-c87c16f360db
[redacted] committed on Jan 3, 2016
24bb27d

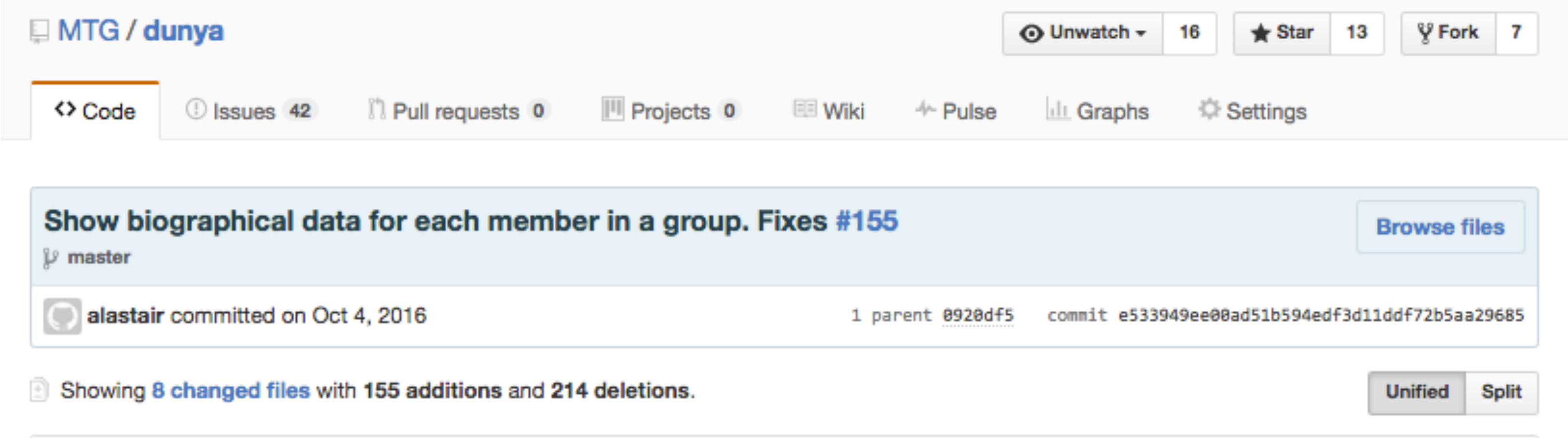
Commits on Jan 1, 2016

Minor corrections for 0b45417b-acb4-4f8a-b180-5ad45be889af
[redacted] committed on Jan 1, 2016
c3a7916

Finished what!? ☹

I can understand what he annotated from the unique document identifiers

- A brief, clear description with a reference to the relevant issue is perfect!

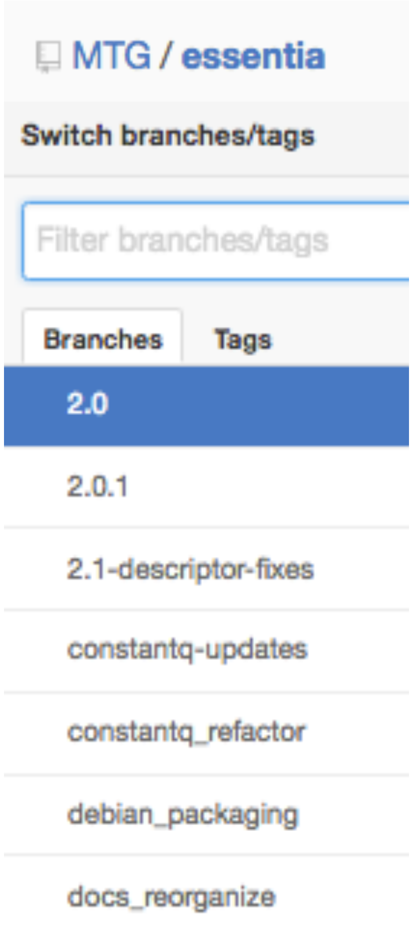


Show biographical data for each member in a group. Fixes #155
master

alastair committed on Oct 4, 2016 1 parent 0920df5 commit e533949ee00ad51b594edf3d11ddf72b5aa29685

Showing 8 changed files with 155 additions and 214 deletions.

Make Branches



MTG / essentia

Switch branches/tags

Filter branches/tags

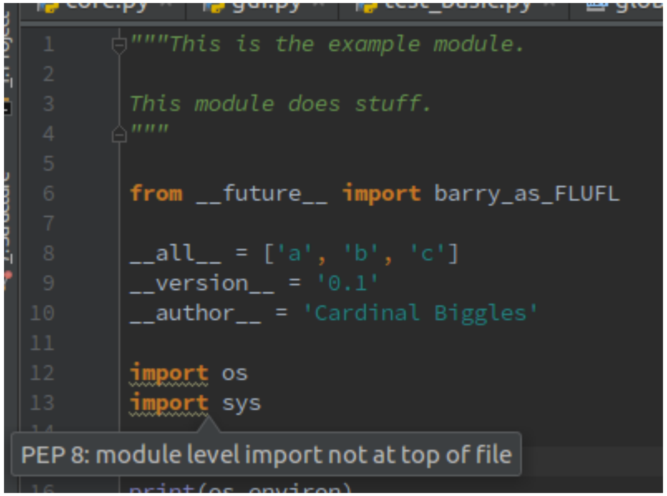
Branches Tags

- 2.0
- 2.0.1
- 2.1-descriptor-fixes
- constantq-updates
- constantq_refactor
- debian_packaging
- docs_reorganize

- When you are working on a conceptual change:
  - Create a branch
  - Introduce the changes to the branch
  - Merge when the solution has matured
- Also good for baking new ideas (without messing up with your main stuff)

Code Quality: Style and Naming Consistency

- Be consistent with the coding style
  - e.g. don’t mix camelCase and underscore_var
- Many languages have a style guideline
  - e.g. *PEP8 for* Python
- Automatic checkers exist
  - *flake8* in Python
- IDEs can point the violations on-the-fly
  - I use *PyCharm* for Python
- Be consistent with namings
  - What does the function “process_myVar” actually do!?
  - It’s a useful to name your modules, variables and organize your code similar to your description in the paper.

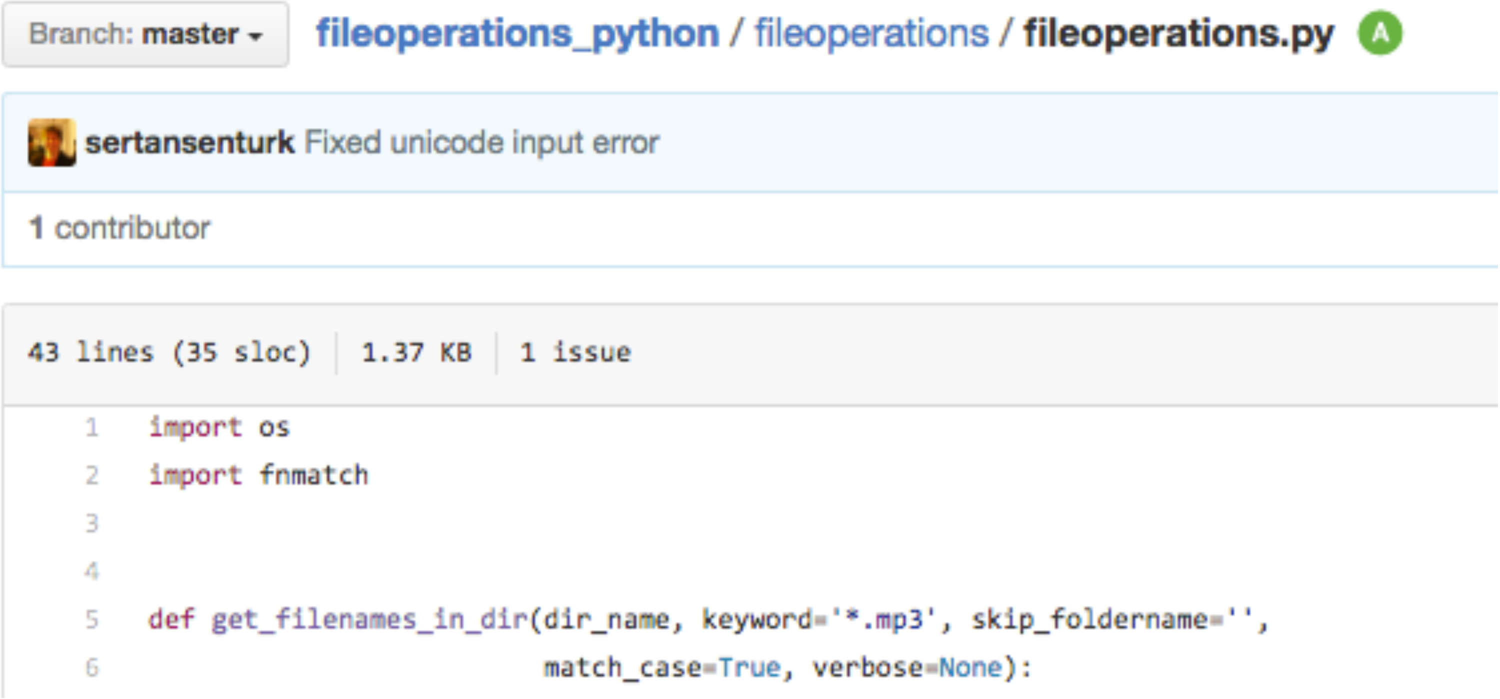


```
"""This is the example module.

This module does stuff.
"""

from __future__ import barry_as_FLUFL

__all__ = ['a', 'b', 'c']
__version__ = '0.1'
__author__ = 'Cardinal Biggles'

import os
import sys
```

PEP 8: module level import not at top of file

```
print(os.environ)
```

Code Quality: Don’t Repeat Yourself



Branch: **master** ▾ **fileoperations_python** / fileoperations / **fileoperations.py** A

sertansenturk Fixed unicode input error

1 contributor

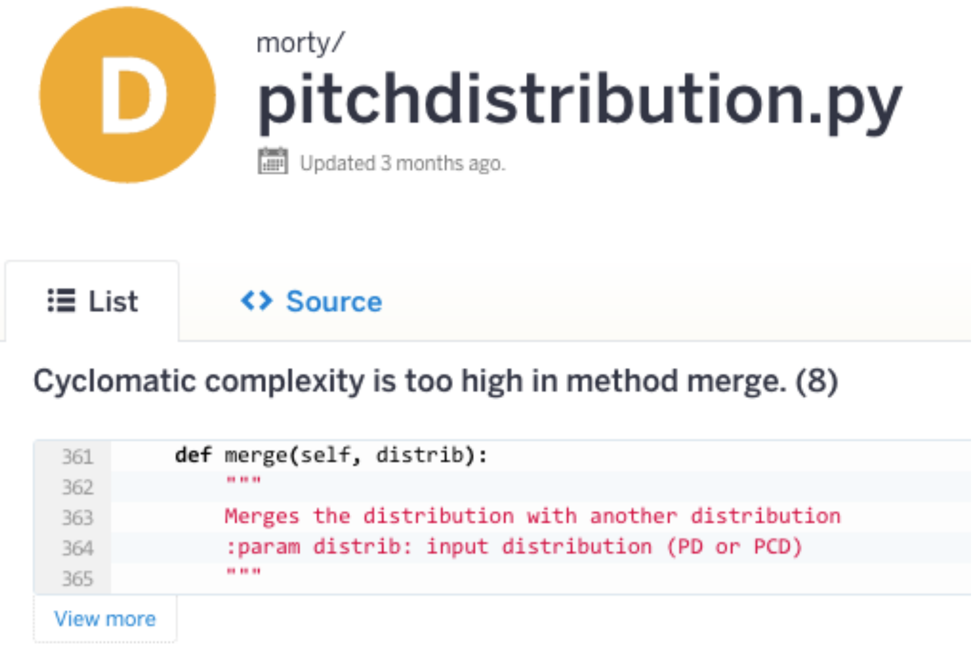
43 lines (35 sloc) | 1.37 KB | 1 issue

```python
import os
import fnmatch


def get_filenames_in_dir(dir_name, keyword='*.mp3', skip_foldername='',
                         match_case=True, verbose=None):
```

- Using this simple repo saves me many minutes every time I start a new experiment! https://github.com/sertansenturk/fileoperations_python

Code Quality: Make Things Simple and Modular

- Modularize
  - If a function is 500 lines, it's time to divide it into smaller pieces
- Make things simple
  - If a function is getting too complex divide into several
  - But don't make them too nested



morty/

pitchdistribution.py

Updated 3 months ago.

Cyclomatic complexity is too high in method merge. (8)

```
    def merge(self, distrib):
        """
        Merges the distribution with another distribution
        :param distrib: input distribution (PD or PCD)
        """
```

https://codeclimate.com
Automatically reports these!

Logging and handling the processes

- Doing these save a lot of time during testing:
  - Handle errors (*try/except* in Python)
  - Have sanity checks (*assert* in Python)
  - Control I/O types
    - Read about "duck typing" in Python (Wikipedia)
  - Log the progress (the *logging* library in Python)
    - Warnings, errors, status etc.

Packaging and Deployment

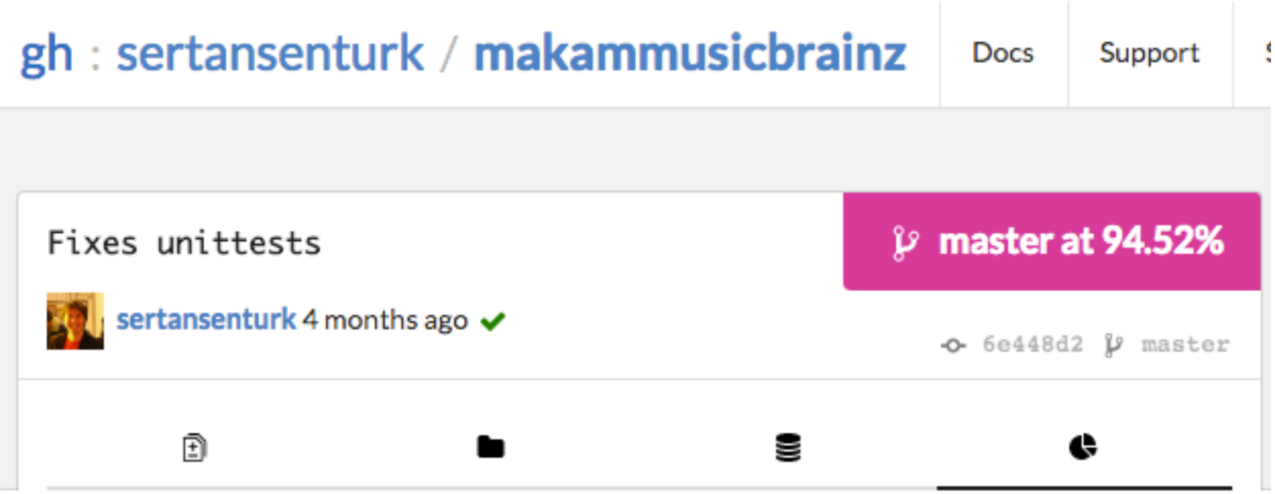
- When possible, put effort to make your code easy to deploy
  - Compile your code in MATLAB, so people can use them via *MATLAB Runtime* without the need of a license
  - Prepare setup.py, so people can install your tool with *pip*
    - Much better if you register to *pypi* (https://pypi.python.org/pypi)
- Clearly state the external requirements
  - Much better, if you can internalize them in the setup
- If the installation is simple, people will prefer your work over others

Write Unit Tests

- Unit tests ensures the code blocks work as intended
  - … and you are not introducing errors during development
  - … and forces you to write better code
- Many languages come with their own easy-to-use schemes
  - Many options in Python (e.g. *nose*, *nose2*, *pytest*)
- Write a unit test per (small) process
  - Use synthetic inputs to cover possible cases
  - Also feed bad cases, where you know things should fail. Check if your code behaves as expected (*try/except* in Python)
- You can also write a single unit test to test the complete run
  - E.g. if it works OK for a hard example, it should work well in general
  - Saves time initially, but usually much dirtier and less useful in the long run

Unit test coverage

- Try to make the unit tests cover all the code base
- Could be computed automatically during unit tests
- Many services exist for visualizing & inspecting the coverage



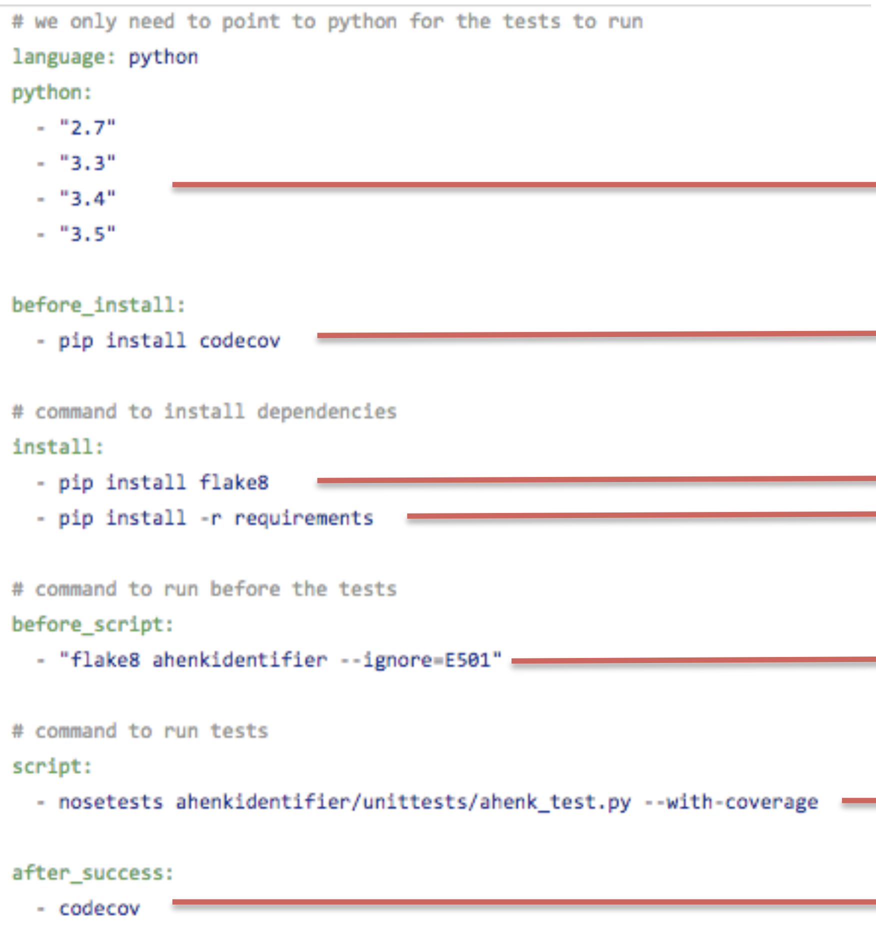
gh : sertansenturk / makammusicbrainz
Docs
Support
Fixes unittests
sertansenturk 4 months ago ✓
master at 94.52%
6e448d2
master

https://codecov.io

Continuous Integration

- Automatically checks if (some of) these are fulfilled:
  - Unit tests
  - Syntax, Style
  - Code Quality
  - Code Coverage
  - …
- They also make sure the installation works seamlessly

Continuous Integration is Easy to Integrate!



```
# we only need to point to python for the tests to run
language: python
python:
  - "2.7"
  - "3.3"
  - "3.4"
  - "3.5"

before_install:
  - pip install codecov

# command to install dependencies
install:
  - pip install flake8
  - pip install -r requirements

# command to run before the tests
before_script:
  - "flake8 ahenkidentifier --ignore=E501"

# command to run tests
script:
  - nosetests ahenkidentifier/unittests/ahenk_test.py --with-coverage

after_success:
  - codecov
```

The Python versions to test

Install Code coverage (https://codecov.io/)

Install syntax checker package

Install required packages

Check syntax (PEP8)

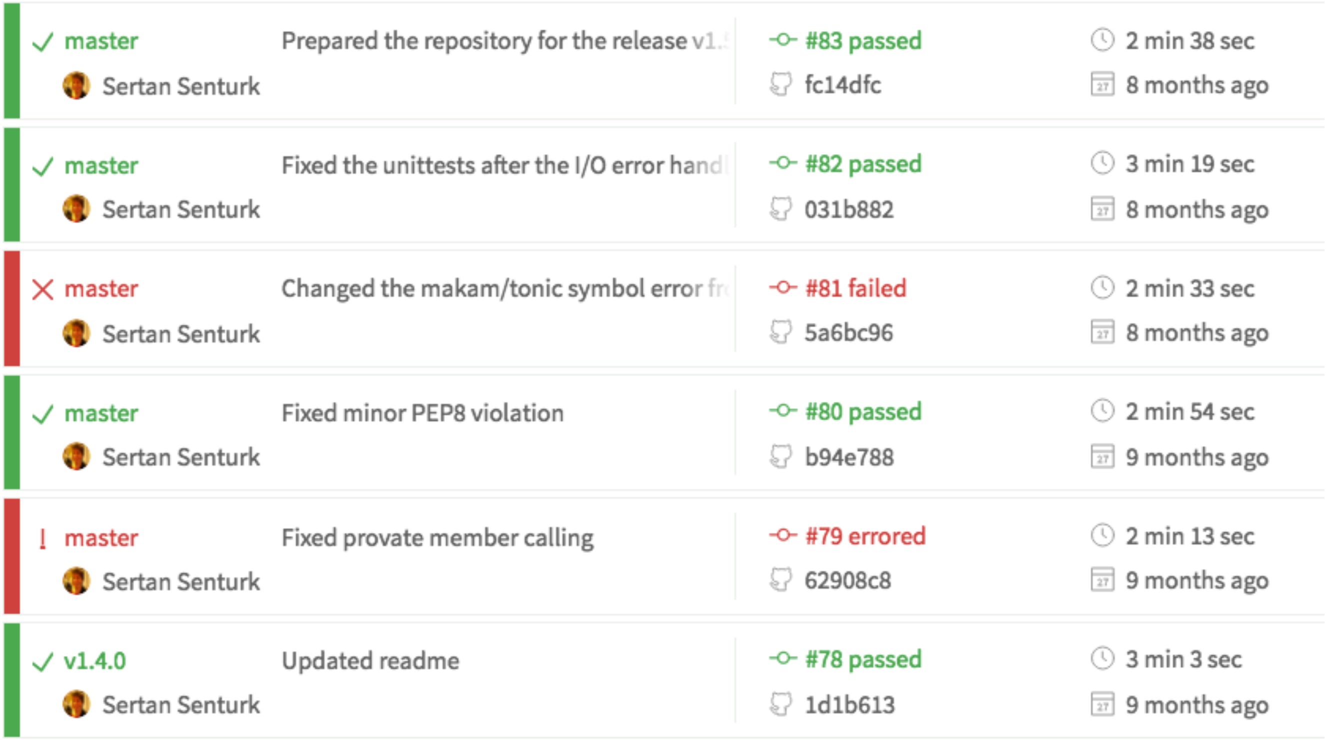
Run unit tests with code coverage

Report code coverage

https://github.com/sertansenturk/ahenkidentifier/blob/v1.5.0/.travis.yml

Continuous Integration is Practical

- You can get notified (via mail, HipChat etc.) immediately, when something breaks!

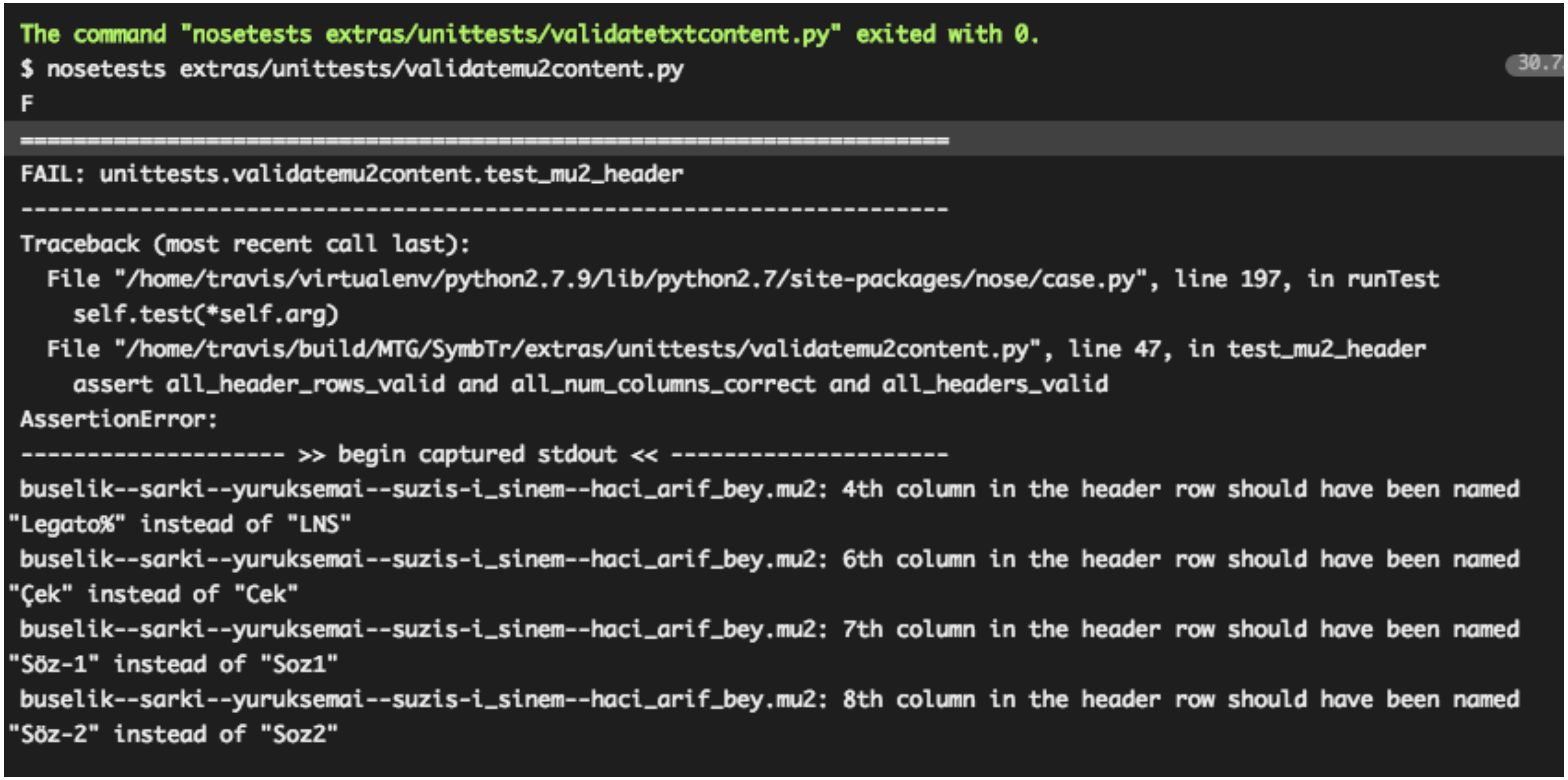


✓ master Sertan Senturk	Prepared the repository for the release v1.	#83 passed fc14dfc	2 min 38 sec 8 months ago
✓ master Sertan Senturk	Fixed the unittests after the I/O error hand	#82 passed 031b882	3 min 19 sec 8 months ago
✗ master Sertan Senturk	Changed the makam/tonic symbol error fr	#81 failed 5a6bc96	2 min 33 sec 8 months ago
✓ master Sertan Senturk	Fixed minor PEP8 violation	#80 passed b94e788	2 min 54 sec 9 months ago
! master Sertan Senturk	Fixed provate member calling	#79 errored 62908c8	2 min 13 sec 9 months ago
✓ v1.4.0 Sertan Senturk	Updated readme	#78 passed 1d1b613	3 min 3 sec 9 months ago

https://travis-ci.org/sertansenturk

Continuous Integration is Practical

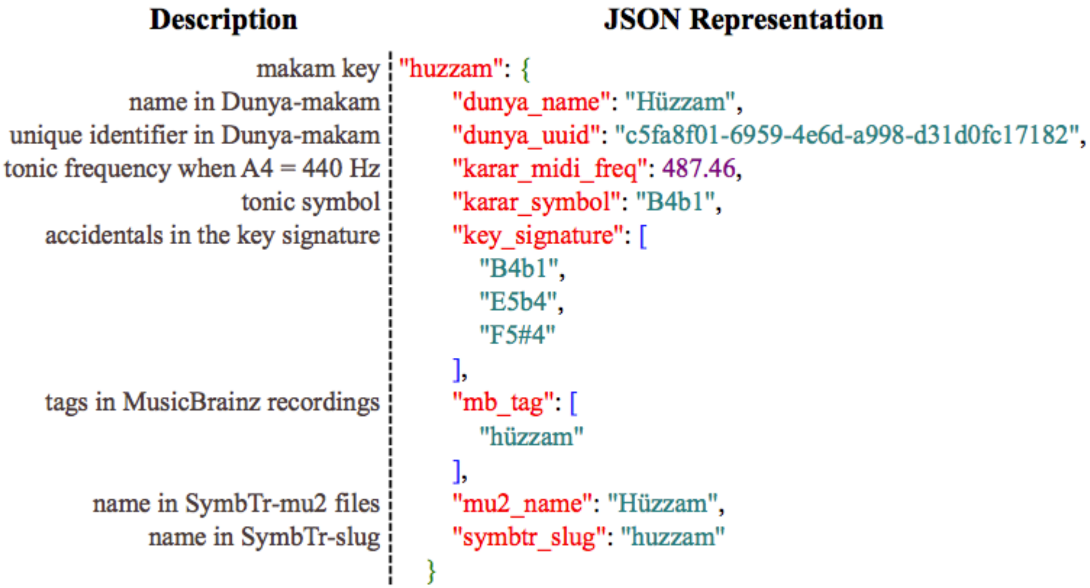
- Descriptive unit tests will save a lot of time in finding the problem(s)
  - Not only effective in fixing code but also in validating data/annotations automatically
  - See the music score example below: https://travis-ci.org/MTG/SymbTr/builds/105130983



```
The command "nosetests extras/unittests/validatetxtcontent.py" exited with 0.
$ nosetests extras/unittests/validatemu2content.py                                         30.7
F
======================================================================
FAIL: unittests.validatemu2content.test_mu2_header
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/travis/virtualenv/python2.7.9/lib/python2.7/site-packages/nose/case.py", line 197, in runTest
    self.test(*self.arg)
  File "/home/travis/build/MTG/SymbTr/extras/unittests/validatemu2content.py", line 47, in test_mu2_header
    assert all_header_rows_valid and all_num_columns_correct and all_headers_valid
AssertionError:
-------------------- >> begin captured stdout << ---------------------
buselik--sarki--yuruksemai--suzis-i_sinem--haci_arif_bey.mu2: 4th column in the header row should have been named
"Legato%" instead of "LNS"
buselik--sarki--yuruksemai--suzis-i_sinem--haci_arif_bey.mu2: 6th column in the header row should have been named
"Çek" instead of "Cek"
buselik--sarki--yuruksemai--suzis-i_sinem--haci_arif_bey.mu2: 7th column in the header row should have been named
"Söz-1" instead of "Soz1"
buselik--sarki--yuruksemai--suzis-i_sinem--haci_arif_bey.mu2: 8th column in the header row should have been named
"Söz-2" instead of "Soz2"
```

Documenting Data

- Clearly explain the data and its organization



Description	JSON Representation
makam key	"huzzam": {
name in Dunya-makam	"dunya_name": "Hüzzam",
unique identifier in Dunya-makam	"dunya_uuid": "c5fa8f01-6959-4e6d-a998-d31d0fc17182",
tonic frequency when A4 = 440 Hz	"karar_midi_freq": 487.46,
tonic symbol	"karar_symbol": "B4b1",
accidentals in the key signature	"key_signature": [
	"B4b1",
	"E5b4",
	"F5#4"
	],
tags in MusicBrainz recordings	"mb_tag": [
	"hüzzam"
	],
name in SymbTr-mu2 files	"mu2_name": "Hüzzam",
name in SymbTr-slug	"symbtr_slug": "huzzam"
	}

- JSON/YAML may be more friendly than tabular formats
  - What does the 15th column mean?

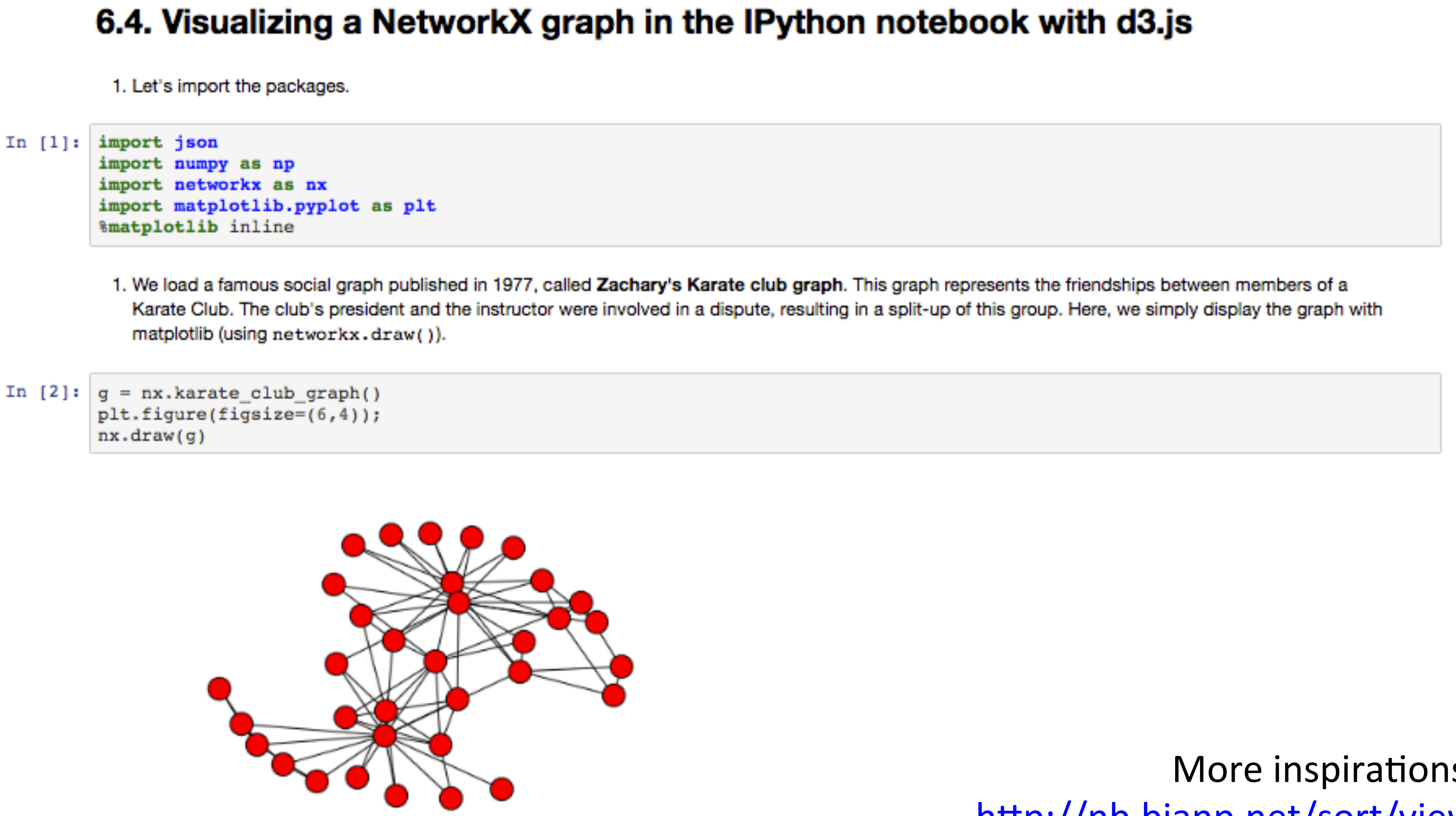
Documenting Code

- For me, the best documentation is a well written code with inline explanations
- Nevertheless, tutorials and manuals always complement!
  - e.g. always provide comprehensive installation instructions (for different environments)
- Use standard documentation styles
  - e.g. Google or Numpy Style Python Docstrings
- Many tools can read these styles and automatically generate good looking documentation
  - e.g. *Sphinx* in Python

Documenting and Organizing Experiments

- Always separate your experimental code from methodology
  - You can reuse them separately later
  - Add the specific release of the (packaged) code to the requirements
- Also, keep the data in a separate repository
  - Import the specific version (*submodules* in git)
- Provide step-by-step instructions to run the experiments
  - A single master script to run them all would be A+
- Ask (beg) a colleague to reproduce your experiment from the material you've published
- If you can show the results interactively, it may complement the narrative in the paper
  - I use *Jupyter notebook* in Python

Jupyter notebook example



6.4. Visualizing a NetworkX graph in the IPython notebook with d3.js

1. Let's import the packages.

```
In [1]: import json
        import numpy as np
        import networkx as nx
        import matplotlib.pyplot as plt
        %matplotlib inline
```

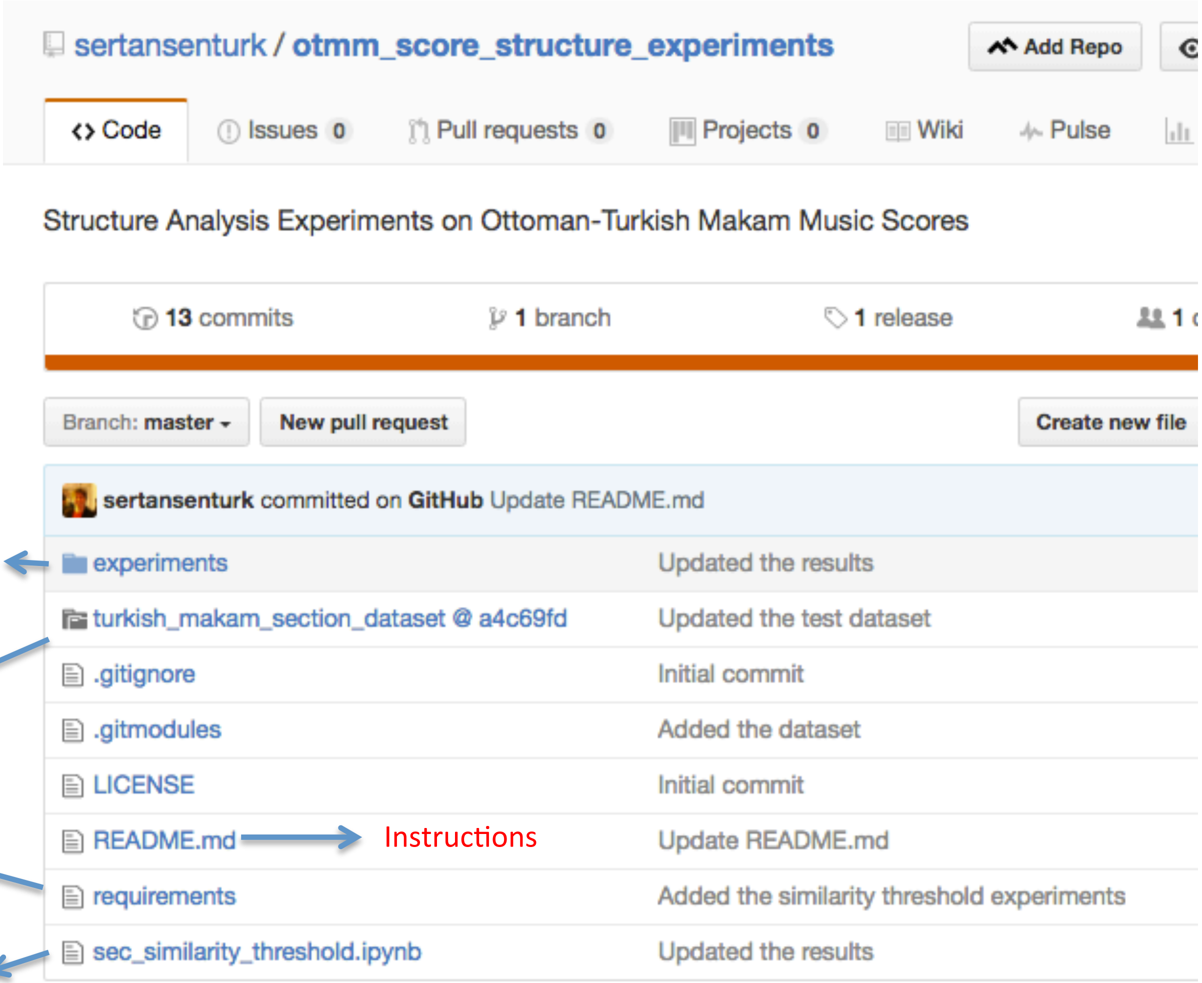
1. We load a famous social graph published in 1977, called **Zachary's Karate club graph**. This graph represents the friendships between members of a Karate Club. The club's president and the instructor were involved in a dispute, resulting in a split-up of this group. Here, we simply display the graph with matplotlib (using `networkx.draw()`).

```
In [2]: g = nx.karate_club_graph()
        plt.figure(figsize=(6,4));
        nx.draw(g)
```

More inspirations at
http://nb.bianp.net/sort/views/

How I (typically) organize my experiments

https://github.com/sertansenturk/otmm_score_structure_experiments/tree/fma_2016



sertansenturk / **otmm_score_structure_experiments**

Structure Analysis Experiments on Ottoman-Turkish Makam Music Scores

13 commits · **1** branch · **1** release · **1**

sertansenturk committed on **GitHub** Update README.md

experiments	Updated the results
turkish_makam_section_dataset @ a4c69fd	Updated the test dataset
.gitignore	Initial commit
.gitmodules	Added the dataset
LICENSE	Initial commit
README.md	Update README.md
requirements	Added the similarity threshold experiments
sec_similarity_threshold.ipynb	Updated the results

Experimental results and evaluation

Dataset (submodule linked to the specific commit)

Package requirements, Incl. my code packaged in a separate repo

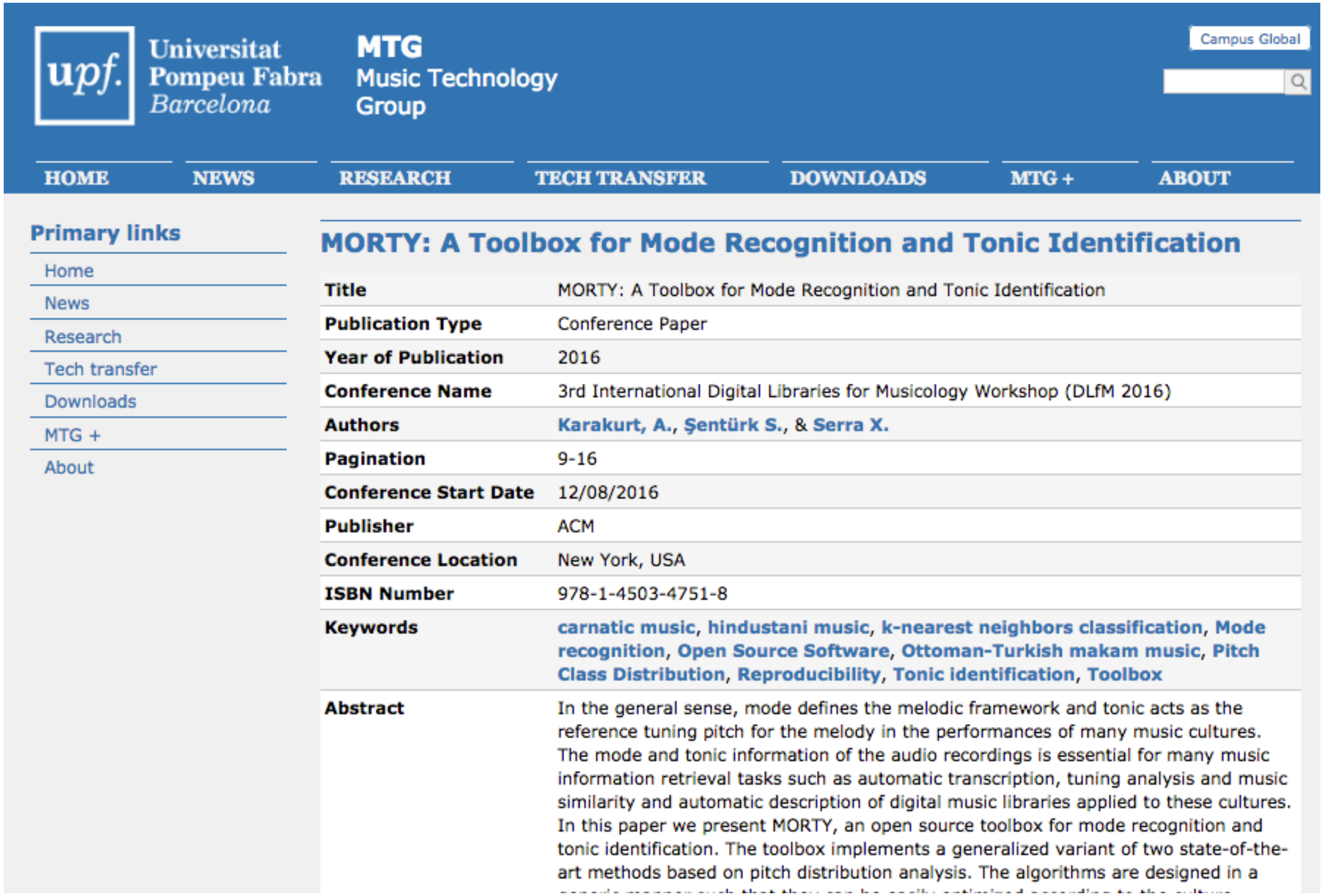
Instructions

Jupyter notebook to run experiments

License

- Always specify the license of your material!
    - People should know the terms to use your product
- Examples:
    - Data, figures etc.: Creative Commons Licenses
    - Code: MIT, BSD, GPL etc.
- Add the information to README of the repository and also to the header of the files

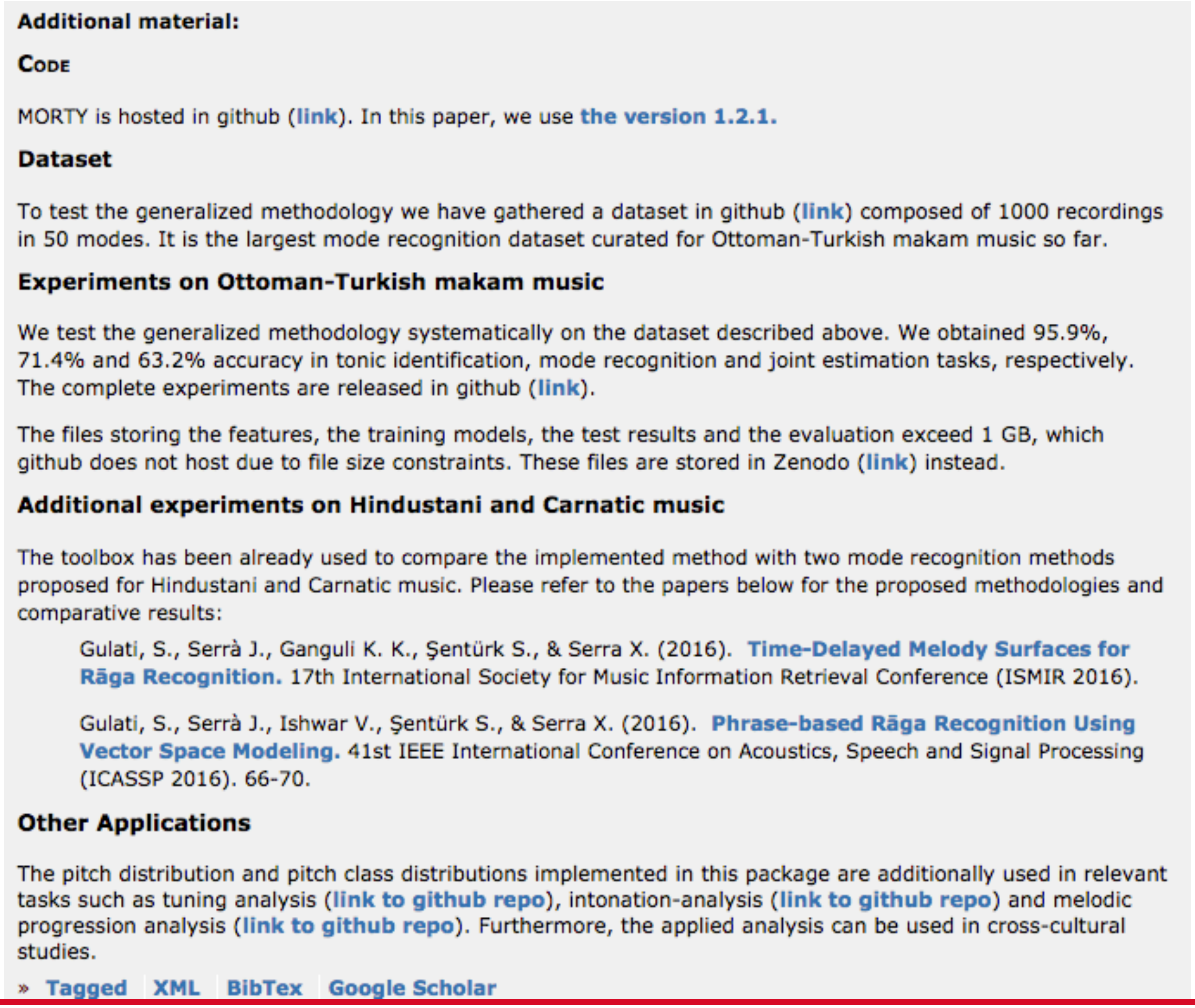
Make Your Publication Visible



MORTY: A Toolbox for Mode Recognition and Tonic Identification

Title	MORTY: A Toolbox for Mode Recognition and Tonic Identification
Publication Type	Conference Paper
Year of Publication	2016
Conference Name	3rd International Digital Libraries for Musicology Workshop (DLfM 2016)
Authors	Karakurt, A., Şentürk S., & Serra X.
Pagination	9-16
Conference Start Date	12/08/2016
Publisher	ACM
Conference Location	New York, USA
ISBN Number	978-1-4503-4751-8
Keywords	carnatic music, hindustani music, k-nearest neighbors classification, Mode recognition, Open Source Software, Ottoman-Turkish makam music, Pitch Class Distribution, Reproducibility, Tonic identification, Toolbox
Abstract	In the general sense, mode defines the melodic framework and tonic acts as the reference tuning pitch for the melody in the performances of many music cultures. The mode and tonic information of the audio recordings is essential for many music information retrieval tasks such as automatic transcription, tuning analysis and music similarity and automatic description of digital music libraries applied to these cultures. In this paper we present MORTY, an open source toolbox for mode recognition and tonic identification. The toolbox implements a generalized variant of two state-of-the-art methods based on pitch distribution analysis. The algorithms are designed in a generic manner such that they can be easily optimized according to the culture

Create Companion Pages



Additional material:

Code

MORTY is hosted in github (**link**). In this paper, we use **the version 1.2.1.**

Dataset

To test the generalized methodology we have gathered a dataset in github (**link**) composed of 1000 recordings in 50 modes. It is the largest mode recognition dataset curated for Ottoman-Turkish makam music so far.

Experiments on Ottoman-Turkish makam music

We test the generalized methodology systematically on the dataset described above. We obtained 95.9%, 71.4% and 63.2% accuracy in tonic identification, mode recognition and joint estimation tasks, respectively. The complete experiments are released in github (**link**).

The files storing the features, the training models, the test results and the evaluation exceed 1 GB, which github does not host due to file size constraints. These files are stored in Zenodo (**link**) instead.

Additional experiments on Hindustani and Carnatic music

The toolbox has been already used to compare the implemented method with two mode recognition methods proposed for Hindustani and Carnatic music. Please refer to the papers below for the proposed methodologies and comparative results:

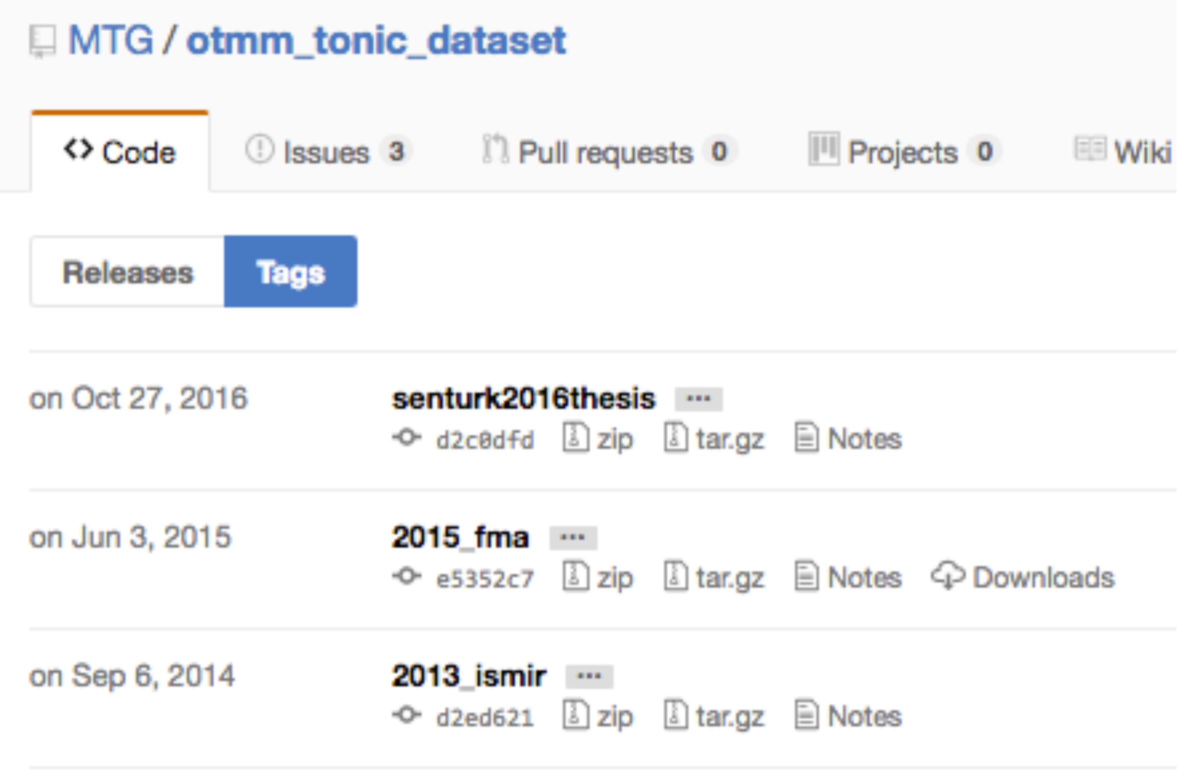
> Gulati, S., Serrà J., Ganguli K. K., Şentürk S., & Serra X. (2016). **Time-Delayed Melody Surfaces for Rāga Recognition.** 17th International Society for Music Information Retrieval Conference (ISMIR 2016).
>
> Gulati, S., Serrà J., Ishwar V., Şentürk S., & Serra X. (2016). **Phrase-based Rāga Recognition Using Vector Space Modeling.** 41st IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2016). 66-70.

Other Applications

The pitch distribution and pitch class distributions implemented in this package are additionally used in relevant tasks such as tuning analysis (**link to github repo**), intonation-analysis (**link to github repo**) and melodic progression analysis (**link to github repo**). Furthermore, the applied analysis can be used in cross-cultural studies.

Release the Relevant Material

- For each publication:



MTG / otmm_tonic_dataset

on Oct 27, 2016 — senturk2016thesis — d2c0dfd zip tar.gz Notes

on Jun 3, 2015 — 2015_fma — e5352c7 zip tar.gz Notes Downloads

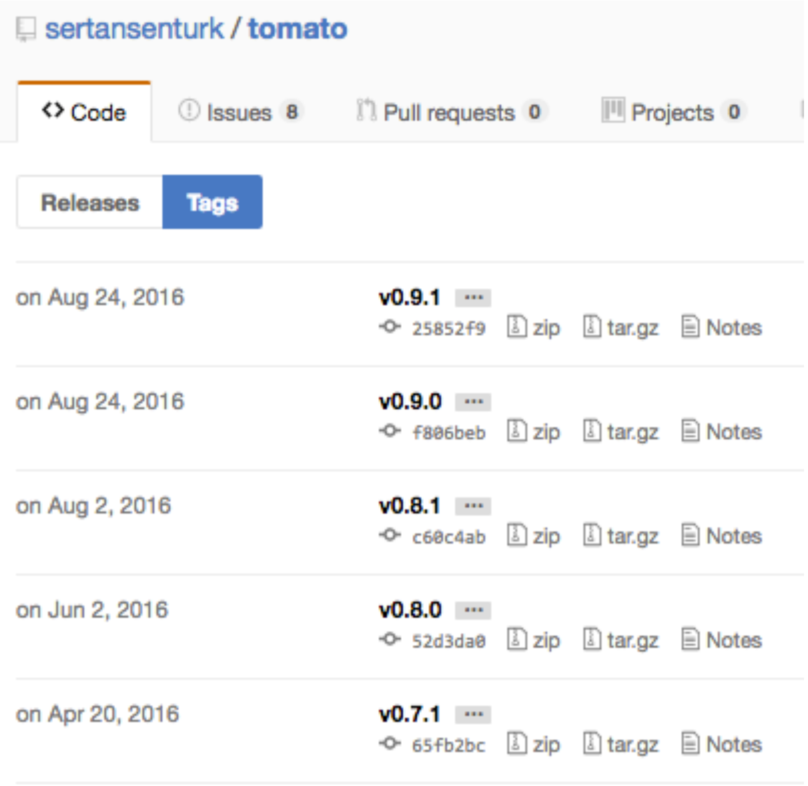
on Sep 6, 2014 — 2013_ismir — d2ed621 zip tar.gz Notes

Release the Code

- For each publication

  AND

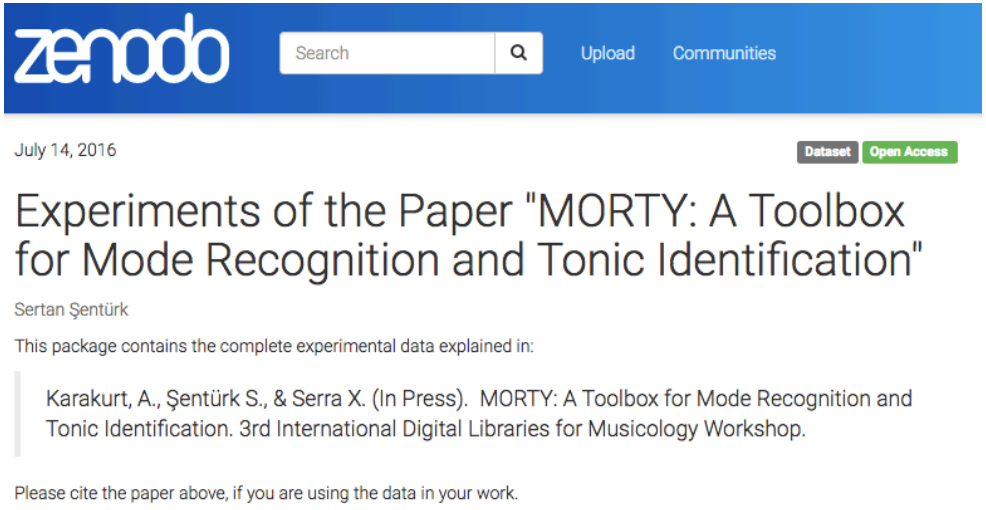
- When a milestone is met
- For bug fixes
- Use semantic versioning
  - MAJOR.MINOR.PATCH (e.g. v2.9.1)
  - MAJOR: incompatible changes
  - MINOR: Compatible changes
  - PATCH: Bug fixes



on Aug 24, 2016	v0.9.1	25852f9	zip tar.gz Notes
on Aug 24, 2016	v0.9.0	f806beb	zip tar.gz Notes
on Aug 2, 2016	v0.8.1	c60c4ab	zip tar.gz Notes
on Jun 2, 2016	v0.8.0	52d3da0	zip tar.gz Notes
on Apr 20, 2016	v0.7.1	65fb2bc	zip tar.gz Notes

Make your Data/Code etc. Citable

- Add a Digital Object Identifier (DOI) to your releases by archiving them in Zenodo



July 14, 2016

Experiments of the Paper "MORTY: A Toolbox for Mode Recognition and Tonic Identification"

Sertan Şentürk

This package contains the complete experimental data explained in:

> Karakurt, A., Şentürk S., & Serra X. (In Press). MORTY: A Toolbox for Mode Recognition and Tonic Identification. 3rd International Digital Libraries for Musicology Workshop.

Please cite the paper above, if you are using the data in your work.

https://zenodo.org/record/57999

- Github integration: https://guides.github.com/activities/citable-code/
- Suitable for your preprints and large data too!

Recap

- Use version control for all of your code, data, experiments and publications
- Keep your code tidy
- Document all your steps clearly
- Freeze (and release) all relevant material by the time you have your paper camera-ready
- Run the experiments in as few steps as possible
- License your work
- Organize all published material properly
- Ask someone to reproduce your work

What do we gain from reproducibility?

- Spend less time/effort on recreating previous research
  - And reusing them on our current work!
- Advance the state of the art
- Improve our visibility
- Have more impact
  - More citations
  - More collaborations
  - Future projects
  - Job offers

Additional Resources

- *Reproducibility in Research* – DTIC-Maria de Maetzu Strategic Program in Universitat Pompeu Fabra (Website)
- *Reproducibility guidelines* by Aurelio Luiz Garcia, prepared within the MdM Strategic Program (Google Document)
- *Licensing models for exploitation of R+D*. Presentation by Malcolm Bain in Universitat Pompeu Fabra (Slides)
- *How Not to Lose Your Code, Your Degree, and Your Future Job.* Presentation by Justin Salamon in New York University (Slides)
- *EECS E6891 – Reproducing Computational Results.* Graduate Course by Dan Ellis & Brian McFee in Columbia University (Website)
- *What is the reproducibility crisis in science and what can we do about it?* Presentation by Dorothy V. M. Bishop in Rhodes Biomedical Association (SlideShare)
- Stodden, V., Leisch, F., & Peng, R. D. (Eds.). (2014). *Implementing reproducible research.* Chapman and Hall/CRC. *(Book)*

upf.

Universitat
Pompeu Fabra
Barcelona

MTG
Music Technology
Group