JAMS: A JSON ANNOTATED MUSIC SPECIFICATION FOR REPRODUCIBLE MIR RESEARCH

Eric J. Humphrey[1,*], **Justin Salamon**[1,2], **Oriol Nieto**[1], **Jon Forsyth**[1],
Rachel M. Bittner[1], **and Juan P. Bello**[1]

[1]Music and Audio Research Lab, New York University, New York
[2]Center for Urban Science and Progress, New York University, New York

ABSTRACT

The continued growth of MIR is motivating more complex annotation data, consisting of richer information, multiple annotations for a given task, and multiple tasks for a given music signal. In this work, we propose JAMS, a JSON-based music annotation format capable of addressing the evolving research requirements of the community, based on the three core principles of simplicity, structure and sustainability. It is designed to support existing data while encouraging the transition to more consistent, comprehensive, well-documented annotations that are poised to be at the crux of future MIR research. Finally, we provide a formal schema, software tools, and popular datasets in the proposed format to lower barriers to entry, and discuss how now is a crucial time to make a concerted effort toward sustainable annotation standards.

1. INTRODUCTION

Music annotations —the collection of observations made by one or more agents about an acoustic music signal— are an integral component of content-based Music Information Retrieval (MIR) methodology, and are necessary for designing, evaluating, and comparing computational systems. For clarity, we define the scope of an annotation as corresponding to time scales at or below the level of a complete song, such as semantic descriptors (tags) or time-aligned chords labels. Traditionally, the community has relied on plain text and custom conventions to serialize this data to a file for the purposes of storage and dissemination, collectively referred to as “lab-files”. Despite a lack of formal standards, lab-files have been, and continue to be, the preferred file format for a variety of MIR tasks, such as beat or onset estimation, chord estimation, or segmentation.

Meanwhile, the interests and requirements of the community are continually evolving, thus testing the practical limitations of lab-files. By our count, there are three unfolding research trends that are demanding more of a given annotation format:

- **Comprehensive annotation data**: Rich annotations, like the Billboard dataset [2], require new, content-specific conventions, increasing the complexity of the software necessary to decode it and the burden on the researcher to use it; such annotations can be so complex, in fact, it becomes necessary to document how to understand and parse the format [5].
- **Multiple annotations for a given task**: The experience of music can be highly subjective, at which point the notion of “ground truth” becomes tenuous. Recent work in automatic chord estimation [8] shows that multiple reference annotations should be embraced, as they can provide important insight into system evaluation, as well as into the task itself.
- **Multiple concepts for a given signal**: Although systems are classically developed to accomplish a single task, there is ongoing discussion toward integrating information across various musical concepts [12]. This has already yielded measurable benefits for the joint estimation of chords and downbeats [9] or chords and segments [6], where leveraging multiple information sources for the same input signal can lead to improved performance.

It has long been acknowledged that lab-files cannot be used to these ends, and various formats and technologies have been previously proposed to alleviate these issues, such as RDF [3], HDF5 [1], or XML [7]. However, none of these formats have been widely embraced by the community. We contend that the weak adoption of any alternative format is due to the combination of several factors. For example, new tools can be difficult, if not impossible, to integrate into a research workflow because of compatibility issues with a preferred development platform or programming environment. Additionally, it is a common criticism that the syntax or data model of these alternative formats is non-obvious, verbose, or otherwise confusing. This is especially problematic when researchers must handle for-

[*] Please direct correspondence to ejhumphrey@nyu.edu

mat conversions. Taken together, the apparent benefits to conversion are outweighed by the tangible costs.

In this paper, we propose a JSON Annotated Music Specification (JAMS) to meet the changing needs of the MIR community, based on three core design tenets: simplicity, structure, and sustainability. This is achieved by combining the advantages of lab-files with lessons learned from previously proposed formats. The resulting JAMS files are human-readable, easy to drop into existing workflows, and provide solutions to the research trends outlined previously. We further address classical barriers to adoption by providing tools for easy use with Python and MATLAB, and by offering an array of popular datasets as JAMS files online. The remainder of this paper is organized as follows: Section 2 identifies three valuable components of an annotation format by considering prior technologies; Section 3 formally introduces JAMS, detailing how it meets these design criteria and describing the proposed specification by example; Section 4 addresses practical issues and concerns in an informal FAQ-style, touching on usage tools, provided datasets, and some practical shortcomings; and lastly, we close with a discussion of next steps and perspectives for the future in Section 5.

2. CORE DESIGN PRINCIPLES

In order to craft an annotation format that might serve the community into the foreseeable future, it is worthwhile to consolidate the lessons learned from both the relative success of lab-files and the challenges faced by alternative formats into a set of principles that might guide our design. With this in mind, we offer that usability, and thus the likelihood of adoption, is a function of three criteria:

2.1 Simplicity

The value of simplicity is demonstrated by lab-files in two specific ways. First, the contents are represented in a format that is intuitive, such that the document model clearly matches the data structure and is human-readable, i.e. uses a lightweight syntax. This is a particular criticism of RDF and XML, which can be verbose compared to plain text. Second, lab-files are conceptually easy to incorporate into research workflows. The choice of an alternative file format can be a significant hurdle if it is not widely supported, as is the case with RDF, or the data model of the document does not match the data model of the programming language, as with XML.

2.2 Structure

It is important to recognize that lab-files developed as a way to serialize tabular data (i.e. arrays) in a language-independent manner. Though lab-files excel at this particular use case, they lack the structure required to encode complex data such as hierarchies or mix different data types, such as scalars, strings, multidimensional arrays, etc. This is a known limitation, and the community has devised a variety of ad hoc strategies to cope with it: folder trees and naming conventions, such as "{X}/{Y}/{Z}.lab", where X, Y, and Z correspond to "artist", "album", and "title", respectively[1]; parsing rules, such as "lines beginning with '#' are to be ignored as comments"; auxiliary websites or articles, decoupled from the annotations themselves, to provide critical information such as syntax, conventions, or methodology. Alternative representations are able to manage more complex data via standardized markup and named entities, such as fields in the case of RDF or JSON, or IDs, attributes and tags for XML.

2.3 Sustainability

Recently in MIR, a more concerted effort has been made toward sustainable research methods, which we see positively impacting annotations in two ways. First, there is considerable value to encoding methodology and metadata directly in an annotation, as doing so makes it easier to both support and maintain the annotation while also enabling direct analyses of this additional information. Additionally, it is unnecessary for the MIR community to develop every tool and utility ourselves; we should instead leverage well-supported technologies from larger communities when possible.

3. INTRODUCING JAMS

So far, we have identified several goals for a music annotation format: a data structure that matches the document model; a lightweight markup syntax; support for multiple annotations, multiple tasks, and rich metadata; easy workflow integration; cross-language compliance; and the use of pre-existing technologies for stability. To find our answer, we need only to look to the web development community, who have already identified a technology that meets these requirements. JavaScript Object Notation (JSON)[2] has emerged as *the* serialization format of the Internet, now finding native support in almost every modern programming language. Notably, it was designed to be maximally efficient and human readable, and is capable of representing complex data structures with little overhead.

JSON is, however, only a syntax, and it is necessary to define formal standards outlining how it should be used for a given purpose. To this end, we define a specification on top of JSON (JAMS), tailored to the needs of MIR researchers.

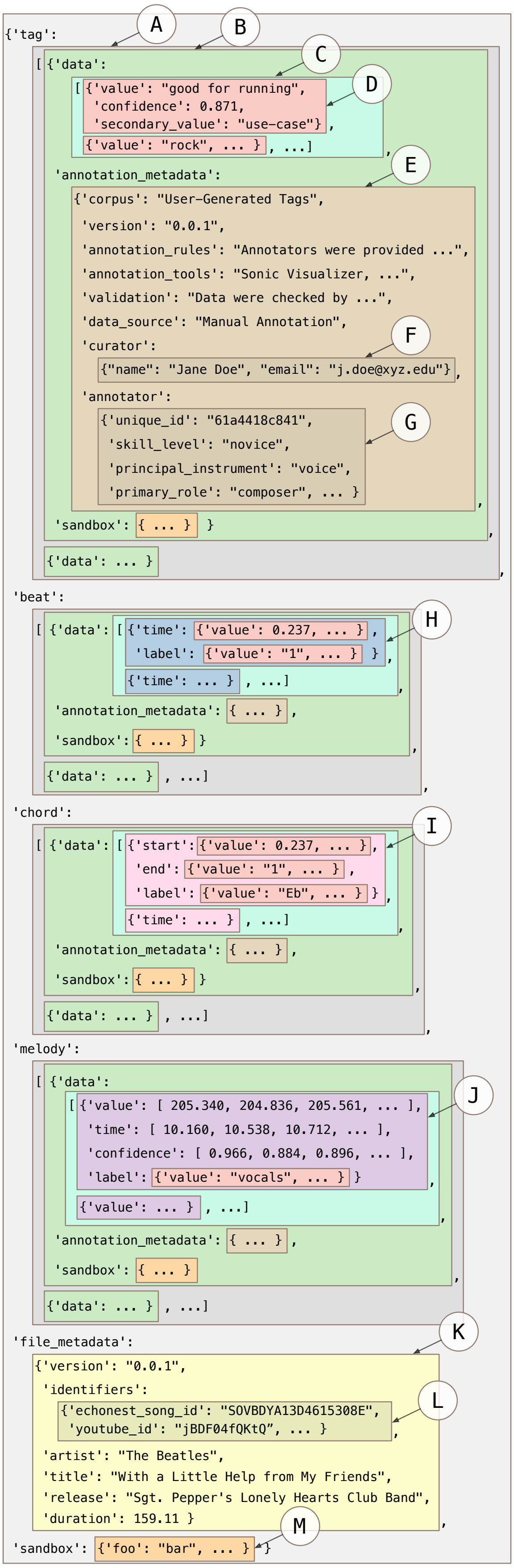
3.1 A Walk-through Example

Perhaps the clearest way to introduce the JAMS specification is by example. Figure 1 provides the contents of a hypothetical JAMS file, consisting of nearly valid[3] JSON syntax and color-coded by concept. JSON syntax will be familiar to those with a background in C-style languages, as it uses square brackets ("[ ]") to denote arrays (alternatively, lists or vectors), and curly brackets ("{ }") to denote

[1] `http://www.isophonics.net/content/reference-annotations`

[2] `http://www.json.org/`

[3] The sole exception is the use of ellipses ("...") as continuation characters, indicating that more information could be included.



```
{'tag':
  [{'data':
     [{'value': "good for running",
       'confidence': 0.871,
       'secondary_value': "use-case"},
      {'value': "rock", ... }, ...],
   'annotation_metadata':
     {'corpus': "User-Generated Tags",
      'version': "0.0.1",
      'annotation_rules': "Annotators were provided ...",
      'annotation_tools': "Sonic Visualizer, ...",
      'validation': "Data were checked by ...",
      'data_source': "Manual Annotation",
      'curator':
        {"name": "Jane Doe", "email": "j.doe@xyz.edu"},
      'annotator':
        {'unique_id': "61a4418c841",
         'skill_level': "novice",
         'principal_instrument': "voice",
         'primary_role': "composer", ... }
     },
   'sandbox': { ... } },
   {'data': ... }
  ],
 'beat':
  [ {'data': [{'time': {'value': 0.237, ... },
               'label': {'value': "1", ... } },
              {'time': ... }, ...],
     'annotation_metadata': { ... },
     'sandbox': { ... } },
    {'data': ... }, ...],
 'chord':
  [ {'data': [{'start': {'value': 0.237, ... },
               'end': {'value': "1", ... },
               'label': {'value': "Eb", ... } },
              {'time': ... }, ...],
     'annotation_metadata': { ... },
     'sandbox': { ... } },
    {'data': ... }, ...],
 'melody':
  [ {'data':
      [{'value': [ 205.340, 204.836, 205.561, ... ],
        'time': [ 10.160, 10.538, 10.712, ... ],
        'confidence': [ 0.966, 0.884, 0.896, ... ],
        'label': {'value': "vocals", ... } },
       {'value': ... }, ...],
     'annotation_metadata': { ... },
     'sandbox': { ... } },
    {'data': ... }, ...],
 'file_metadata':
  {'version': "0.0.1",
   'identifiers':
     {'echonest_song_id': "SOVBDYA13D4615308E",
      'youtube_id': "jBDF04fQKtQ", ... },
   'artist': "The Beatles",
   'title': "With a Little Help from My Friends",
   'release': "Sgt. Pepper's Lonely Hearts Club Band",
   'duration': 159.11 },
 'sandbox': {'foo': "bar", ... }}
```

Figure 1. Diagram illustrating the structure of the JAMS specification.

objects (alternatively, dictionaries, structs, or hash maps). Defining some further conventions for the purpose of illustration, we use single quotes to indicate field names, italics when referring to concepts, and consistent colors for the same data structures. Using this diagram, we will now step through the hierarchy, referring back to relevant components as concepts are introduced.

3.1.1 The JAMS Object

A JAMS file consists of one top-level object, indicated by the outermost bounding box. This is the primary container for all information corresponding to a music signal, consisting of several task-array pairs, an object for file_metadata, and an object for sandbox. A task-array is a list of *annotations* corresponding to a given task name, and may contain zero, one, or many annotations for that task. The format of each array is specific to the kind of annotations it will contain; we will address this in more detail in Section 3.1.2.

The file_metadata object (K) is a dictionary containing basic information about the music signal, or file, that was annotated. In addition to the fields given in the diagram, we also include an unconstrained identifiers object (L), for storing unique identifiers in various namespaces, such as the EchoNest or YouTube. Note that we purposely do not store information about the recording's audio encoding, as a JAMS file is format-agnostic. In other words, we assume that any sample rate or perceptual codec conversions will have no effect on the annotation, within a practical tolerance.

Lastly, the JAMS object also contains a sandbox, an unconstrained object to be used as needed. In this way, the specification carves out such space for any unforeseen or otherwise relevant data; however, as the name implies, no guarantee is made as to the existence or consistency of this information. We do this in the hope that the specification will not be unnecessarily restrictive, and that commonly "sandboxed" information might become part of the specification in the future.

3.1.2 Annotations

An *annotation* (B) consists of all the information that is provided by a single annotator about a single task for a single music signal. Independent of the task, an annotation comprises three sub-components: an array of objects for data (C), an annotation_metadata object (E), and an annotation-level sandbox. For clarity, a task-array (A) may contain multiple annotations (B).

Importantly, a data array contains the primary annotation information, such as its chord sequence, beat locations, etc., and is the information that would normally be stored in a lab-file. Though all data containers are functionally equivalent, each may consist of only one object type, specific to the given task. Considering the different types of musical attributes annotated for MIR research, we divide them into four fundamental categories:

1. Attributes that exist as a single *observation* for the entire music signal, e.g. tags.

2. Attributes that consist of sparse *events* occurring at specific times, e.g. beats or onsets.

3. Attributes that span a certain time *range*, such as chords or sections.

4. Attributes that comprise a dense *time series*, such as discrete-time fundamental frequency values for melody extraction.

These four types form the most atomic data structures, and will be revisited in greater detail in Section 3.1.3. The important takeaway here, however, is that data arrays are not allowed to mix fundamental types.

Following [10], an `annotation_metadata` object is defined to encode information about what has been annotated, who created the annotations, with what tools, etc. Specifically, *corpus* provides the name of the dataset to which the annotation belongs; *version* tracks the version of this particular annotation; *annotation_rules* describes the protocol followed during the annotation process; *annotation_tools* describes the tools used to create the annotation; *validation* specifies to what extent the annotation was verified and is reliable; *data_source* details how the annotation was obtained, such as manual annotations, online aggregation, game with a purpose, etc.; *curator* (F) is itself an object with two subfields, *name* and *email*, for the contact person responsible for the annotation; and *annotator* (G) is another unconstrained object, which is intended to capture information about the source of the annotation. While complete metadata are strongly encouraged in practice, currently only *version* and *curator* are mandatory in the specification.

3.1.3 Datatypes

Having progressed through the JAMS hierarchy, we now introduce the four atomic data structures, out of which an annotation can be constructed: *observation*, *event*, *range* and *time series*. For clarity, the *data* array (A) of a `tag` annotation is a list of *observation* objects; the *data* array of a `beat` annotation is a list of *event* objects; the *data* array of a `chord` annotation is a list of *range* objects; and the *data* array of a `melody` annotation is a list of *time series* objects. The current space of supported tasks is provided in Table 1.

Of the four types, an *observation* (D) is the most atomic, and used to construct the other three. It is an object that has one primary field, `value`, and two optional fields, `confidence` and `secondary_value`. The `value` and `secondary_value` fields may take any simple primitive, such as a string, numerical value, or boolean, whereas the `confidence` field stores a numerical confidence estimate for the observation. A secondary value field is provided for flexibility in the event that an observation requires an additional level of specificity, as is the case in hierarchical segmentation [11].

An *event* (H) is useful for representing musical attributes that occur at sparse moments in time, such as beats or onsets. It is a container that holds two *observations*, `time` and `label`. Referring to the first beat annotation in the diagram, the `value` of *time* is a scalar quantity (`0.237`), whereas the `value` of `label` is a string (`'1'`), indicating metrical position.

observation	*event*	*range*	*time series*

`tag`
`genre`
`mood`
	`beat`
	`onset`
		`chord`
		`segment`
		`key`
		`note`
		`source`
			`melody`
			`pitch`
			`pattern`

Table 1. Currently supported tasks and types in JAMS.

A *range* (I) is useful for representing musical attributes that span an interval of time, such as chords or song segments (e.g. intro, verse, chorus). It is an object that consists of three *observations*: `start`, `end`, and `label`.

The *time series* (J) atomic type is useful for representing musical attributes that are continuous in nature, such as fundamental frequency over time. It is an object composed of four elements: `value`, `time`, `confidence` and `label`. The first three fields are arrays of numerical values, while `label` is an *observation*.

3.2 The JAMS Schema

The description in the previous sections provides a high-level understanding of the proposed specification, but the only way to describe it without ambiguity is through formal representation. To accomplish this, we provide a JSON schema [4], a specification itself written in JSON that uses a set of reserved keywords to define valid data structures. In addition to the expected contents of the JSON file, the schema can specify which fields are required, which are optional, and the type of each field (e.g. numeric, string, boolean, array or object). A JSON schema is concise, precise, and human readable.

Having defined a proper JSON schema, an added benefit of JAMS is that a validator can verify whether or not a piece of JSON complies with a given schema. In this way, researchers working with JAMS files can easily and confidently test the integrity of a dataset. There are a number of JSON schema validator implementations freely available online in a variety of languages, including Python, Java, C, JavaScript, Perl, and more. The JAMS schema is included in the public software repository (cf. Section 4), which also provides a static URL to facilitate directly accessing the schema from the web within a workflow.

4. JAMS IN PRACTICE

While we contend that the use and continued development of JAMS holds great potential for the many reasons outlined previously, we acknowledge that specifications and standards are myriad, and it can be difficult to ascertain the benefits or shortcomings of one's options. In the interest of encouraging adoption and the larger discussion of

[4] `http://json-schema.org/`

standards in the field, we would like to address practical concerns directly.

4.1 How is this any different than *X*?

The biggest advantage of JAMS is found in its capacity to consistently represent rich information with no additional effort from the parser and minimal markup overhead. Compared to XML or RDF, JSON parsers are extremely fast, which has contributed in no small part to its widespread adoption. These efficiency gains are coupled with the fact that JAMS makes it easier to manage large data collections by keeping all annotations for a given song in the same place.

4.2 What kinds of things can I do with JAMS that I can't already do with *Y*?

JAMS can enable much richer evaluation by including multiple, possibly conflicting, reference annotations and directly embedding information about an annotation's origin. A perfect example of this is found in the Rock Corpus Dataset [4], consisting of annotations by two expert musicians: one, a guitarist, and the other, a pianist. Sources of disagreement in the transcriptions often stem from differences of opinion resulting from familiarity with their principal instrument, where the voicing of a chord that makes sense on piano is impossible for a guitarist, and vice versa. Similarly, it is also easier to develop versatile MIR systems that combine information across tasks, as that information is naturally kept together.

Another notable benefit of JAMS is that it can serve as a data representation for algorithm outputs for a variety of tasks. For example, JAMS could simplify MIREX submissions by keeping all machine predictions for a given team together as a single submission, streamlining evaluations, where the annotation sandbox and annotator metadata can be used to keep track of algorithm parameterizations. This enables the comparison of many references against many algorithmic outputs, potentially leading to a deeper insight into a system's performance.

4.3 So how would this interface with my workflow?

Thanks to the widespread adoption of JSON, the vast majority of languages already offer native JSON support. In most cases, this means it is possible to go from a JSON file to a programmatic data structure in your language of choice in a single line of code using tools you didn't have to write. To make this experience even simpler, we additionally provide two software libraries, for Python and MATLAB. In both instances, a lightweight software wrapper is provided to enable a seamless experience with JAMS, allowing IDEs and interpreters to make use of autocomplete and syntax checking. Notably, this allows us to provide convenience functionality for creating, populating, and saving JAMS objects, for which examples and sample code are provided with the software library [5] .

4.4 What datasets are already JAMS-compliant?

To further lower the barrier to entry and simplify the process of integrating JAMS into a pre-existing workflow, we have collected some of the more popular datasets in the community and converted them to the JAMS format, linked via the public repository. The following is a partial list of converted datasets: Isophonics (beat, chord, key, segment); Billboard (chord); SALAMI (segment, pattern); RockCorpus (chord, key); tmc323 (chords); Cal500 (tag); Cal10k (tag); ADC04 (melody); and MIREX05 (melody).

4.5 Okay, but *my* data is in a different format – now what?

We realize that it is impractical to convert every dataset to JAMS, and provide a collection of Python scripts that can be used to convert lab-files to JAMS. In lieu of direct interfaces, alternative formats can first be converted to lab-files and translated to JAMS thusly.

4.6 My MIR task doesn't really fit with JAMS.

That's not a question, but it is a valid point and one worth discussing. While this first iteration of JAMS was designed to be maximally useful across a variety of tasks, there are two broad reasons why JAMS might not work for a given dataset or task. One, a JAMS annotation only considers information at the temporal granularity of a single audio file and smaller, independently of all other audio files in the world. Therefore, extrinsic relationships, such as cover songs or music similarity, won't directly map to the specification because the concept is out of scope.

The other, more interesting, scenario is that a given use case requires functionality we didn't plan for and, as a result, JAMS doesn't yet support. To be perfectly clear, the proposed specification is exactly that –a proposal– and one under active development. Born out of an internal need, this initial release focuses on tasks with which the authors are familiar, and we realize the difficulty in solving a global problem in a single iteration. As will be discussed in greater detail in the final section, the next phase on our roadmap is to solicit feedback and input from the community at large to assess and improve upon the specification. If you run into an issue, we would love to hear about your experience.

4.7 This sounds promising, but nothing's perfect. There must be shortcomings.

Indeed, there are two practical limits that should be mentioned. Firstly, JAMS is not designed for features or signal level statistics. That said, JSON is still a fantastic, cross-language syntax for serializing data, and may further serve a given workflow. As for practical concerns, it is a known issue that parsing large JSON objects can be slow in MATLAB. We've worked to make this no worse than reading current lab-files, but speed and efficiency are not touted benefits of MATLAB. This may become a bigger issue as JAMS files become more complete over time, but we are

[5] `https://github.com/urinieto/jams`

actively exploring various engineering solutions to address this concern.

5. DISCUSSION AND FUTURE PERSPECTIVES

In this paper, we have proposed a JSON format for music annotations to address the evolving needs of the MIR community by keeping multiple annotations for multiple tasks alongside rich metadata in the same file. We do so in the hopes that the community can begin to easily leverage this depth of information, and take advantage of ubiquitous serialization technology (JSON) in a consistent manner across MIR. The format is designed to be intuitive and easy to integrate into existing workflows, and we provide software libraries and pre-converted datasets to lower barriers to entry.

Beyond practical considerations, JAMS has potential to transform the way researchers approach and use music annotations. One of the more pressing issues facing the community at present is that of dataset curation and access. It is our hope that by associating multiple annotations for multiple tasks to an audio signal with retraceable metadata, such as identifiers or URLs, it might be easier to create freely available datasets with better coverage across tasks. Annotation tools could serve music content found freely on the Internet and upload this information to a common repository, ideally becoming something like a Freebase[6] for MIR. Furthermore, JAMS provides a mechanism to handle multiple concurrent perspectives, rather than forcing the notion of an objective truth.

Finally, we recognize that any specification proposal is incomplete without an honest discussion of feasibility and adoption. The fact remains that JAMS arose from the combination of needs within our group and an observation of wider applicability. We have endeavored to make the specification maximally useful with minimal overhead, but appreciate that community standards require iteration and feedback. This current version is not intended to be the definitive answer, but rather a starting point from which the community can work toward a solution as a collective. Other professional communities, such as the IEEE, convene to discuss standards, and perhaps a similar process could become part of the ISMIR tradition as we continue to embrace the pursuit of reproducible research practices.

6. REFERENCES

[1] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proc. of the 12th International Society for Music Information Retrieval Conference*, pages 591–596, 2011.

[2] John Ashley Burgoyne, Jonathan Wild, and Ichiro Fujinaga. An expert ground truth set for audio chord recognition and music analysis. In *Proc. of the 12th International Society for Music Information Retrieval Conference*, pages 633–638, 2011.

[3] Chris Cannam, Christian Landone, Mark B Sandler, and Juan Pablo Bello. The sonic visualiser: A visualisation platform for semantic descriptors from musical signals. In *Proc. of the 7th International Society for Music Information Retrieval Conference*, pages 324–327, 2006.

[4] Trevor De Clercq and David Temperley. A corpus analysis of rock harmony. *Popular Music*, 30(1):47–70, 2011.

[5] W Bas de Haas and John Ashley Burgoyne. Parsing the billboard chord transcriptions. *University of Utrecht, Tech. Rep*, 2012.

[6] Matthias Mauch, Katy Noland, and Simon Dixon. Using musical structure to enhance automatic chord transcription. In *Proc. of the 10th International Society for Music Information Retrieval Conference*, pages 231–236, 2009.

[7] Cory McKay, Rebecca Fiebrink, Daniel McEnnis, Beinan Li, and Ichiro Fujinaga. Ace: A framework for optimizing music classification. In *Proc. of the 6th International Society for Music Information Retrieval Conference*, pages 42–49, 2005.

[8] Yizhao Ni, Matthew McVicar, Raul Santos-Rodriguez, and Tijl De Bie. Understanding effects of subjectivity in measuring chord estimation accuracy. *Audio, Speech, and Language Processing, IEEE Transactions on*, 21(12):2607–2615, 2013.

[9] Hélène Papadopoulos and Geoffroy Peeters. Joint estimation of chords and downbeats from an audio signal. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19(1):138–152, 2011.

[10] G. Peeters and K. Fort. Towards a (better) definition of annotated MIR corpora. In *Proc. of the 13th International Society for Music Information Retrieval Conference*, pages 25–30, Porto, Portugal, Oct. 2012.

[11] Jordan Bennett Louis Smith, John Ashley Burgoyne, Ichiro Fujinaga, David De Roure, and J Stephen Downie. Design and creation of a large-scale database of structural annotations. In *Proc. of the 12th International Society for Music Information Retrieval Conference*, pages 555–560, 2011.

[12] Emmanuel Vincent, Stanislaw A Raczynski, Nobutaka Ono, Shigeki Sagayama, et al. A roadmap towards versatile mir. In *Proc. of the 11th International Society for Music Information Retrieval Conference*, pages 662–664, 2010.

[6] http://www.freebase.com