Pierre Alexandre Tremblay, Gerard Roma, and Owen Green
Centre for Research in New Music
University of Huddersfield
Queensgate Campus, Huddersfield,
HD1 3DH, United Kingdom
{p.a.tremblay, g.roma, o.green}@hud.ac.uk

Enabling Programmatic Data Mining as Musicking: The Fluid Corpus Manipulation Toolkit

Abstract: This article presents a new software toolbox to enable programmatic mining of sound banks for musicking and musicking-driven research. The toolbox is available for three popular creative coding environments currently used by "techno-fluent" musicians. The article describes the design rationale and functionality of the toolbox and its ecosystem, then the development methodology—several versions of the toolbox have been seeded to early adopters who have, in turn, contributed to the design. Examples of these early usages are presented, and we describe some observed musical affordances of the proposed approach to the exploration and manipulation of music corpora, as well as the main roadblocks encountered. We finally reflect on a few emerging themes for the next steps in building a community around critical programmatic mining of sound banks.

Developments in general computing continue to be inspired by the possibilities of exploiting large amounts of data. For "musicking" (Small 1998), taken as music making in its widest, most diverse, and inclusive sense, new data-driven paradigms are promising but still challenging, owing to the diversity and size of sound collections amassed by musicians, as well as to the context-dependent nature of questions that determine the paths of musical creativity.

Notably, although there has been much research around the potential of isolated technologies or techniques in a musical context, there has been much less detailing or comparing of musical strategies and tactics arising from sustained practice. The Fluid Corpus Manipulation project (FluCoMa) aims to help bridge this gap by making available cross-platform technologies in tandem with supporting resources and community, all focused on discovering and developing complete creative workflows for audio corpora with data-driven techniques.

In previous work (Tremblay et al. 2019), we introduced tools for decomposing and analyzing sounds to facilitate the creation of audio corpora and the extraction of audio descriptors from within the most popular musical creative coding environments (CCEs): Max, Pure Data (Pd), and SuperCollider. This leaves open the challenge of enabling musicking and musicking-driven research using the resulting corpora and data.

In this article we present the second iteration of our toolbox, in which we focus on exploring, interacting with, and manipulating audio corpora with a framework of tools for organizing and learning from data. Our aim is to provide this combined functionality in a single package for each CCE with enough commonality in syntax and concepts to enable discussion and exchange between a range of musicians, in the interest of supporting a broad community for future research.

In the next section, we review existing tools for music creation based on audio corpora and note some limitations with respect to the musical workflows they imply. We follow with a description of the aims and content of our toolbox, followed by a section describing examples of workflows enabled by the toolbox from our community of early adopters. We then dedicate a section to offering some insights into our design process and early observations on the project's impact. As described previously (Green, Tremblay, and Roma 2018), our methodology is rooted in a pluralist and cross-disciplinary view of techno-fluent musicking. Here, we focus on some emerging themes around knowledge transfer, interface granularity, and points of entanglement. We finish the article with some anticipated next steps in the Fluid Corpus Manipulation project, with the view of augmenting the toolset, as well as strengthening and diversifying an emerging community.

Computer Music Journal, 45:2, pp. 9–23, Summer 2021
doi:10.1162/COMJ_a_00600

Related Work

There is an extensive body of prior work that provides some of the tools and techniques in which we are interested. Although none of this work aligns precisely with our focus on developing a framework aimed squarely at flexible music making with audio corpora, it has been a rich source of learning and motivation. Our design has also been influenced by the wealth of libraries and learning materials in data science communities, notably around the Python language's scikit-learn library (Pedregosa et al. 2011), and their reflections on interface (Buitinck et al. 2013).

One category of existing work focuses on specific forms of musical interaction. AudioGuide (Hackbarth, Schnell, and Schwarz 2010) and Orchidea (Carpentier et al. 2012) are geared towards computer-assisted orchestration. Meanwhile, CataRT (Schwarz et al. 2006) and the more recent AudioStellar (Garber, Ciccola, and Amusategui 2021) offer an intuitive interface for corpus exploration through snippets arranged in a 2-D space. In these cases, the very specific focus means that the system presents itself as more of a black box than is our aim in the current work, insofar as certain decisions about workflow, analysis, presentation, and general interface are already taken and fixed. This limits the creative-coding affordances of these packages: It makes exploration of interaction, customization, and integration within the CCE more difficult, if at all possible.

A second category is work that makes available particular algorithms or tools in isolation, but not as part of a framework that also supports gathering, exploring, and refining data as part of an overall musical project. Here one could include the Wekinator application (Fiebrink and Cook 2010), the Max packages ml.lib (Bullock and Momeni 2015) and ml.star (Smith and Deal 2014), and certain SuperCollider extensions (called *quarks*) such as the KDTree and KMeans quarks. Most of these solutions are focused more on gestural data streams and offer neither facilities for exploring large corpora nor for dynamic creative coding between various complementary algorithms. Conversely, toolsets that enable handling and exploring data within CCEs have been proposed, but without a focus on audio and machine-learning algorithms. For instance, the Bach family of extensions for Max (Agostini and Ghisi 2012) has recently been enriched with a suite of objects that provides alternatives to Bach's core focus on Western common notation (Ghisi and Agon 2016).

Finally, two prior contributions align more closely with the proposed work, and thus warrant more extensive discussion: the SCMIR package for SuperCollider (Collins 2011), and the MuBu and Friends package for Max (Schnell et al. 2009). Both of these offer some version of an end-to-end pipeline for gathering, analyzing, and exploring data, although neither addresses the additional challenge of targeting multiple CCEs.

The SCMIR package provides a complete pipeline from audio feature extraction to a range of algorithms and representations, along with some supporting machinery for conditioning data. The lack of a native plug-in API for sclang (the language side of SuperCollider) is worked around by using external programs as "pseudoplug-ins." Although the initial focus for SCMIR was analysis, it also enables creative work with analysis data and algorithms. Our toolbox similarly enables complete pipelines, but all the computation is implemented in server-side SuperCollider plug-ins. This provides a more natural and granular integration with the SuperCollider environment.

MuBu and Friends likewise offers a framework with pipelines from audio feature extraction (via PiPo, cf. Schnell et al. 2017), through to a suite of algorithms for exploring and learning from data. Its focus is more general than our proposed toolbox, insofar as gestural data is also an explicit concern for MuBu, although less attention is given to the framework's place in an overall creative workflow. At the heart of framework is the MuBu (multibuffer) object itself, which extends the familiar paradigm of the audio buffer with labels, different datatypes, time tags, and other useful facilities. Our experience has been that this richness can be difficult to grasp, however, especially for newcomers, and that interoperation with the wider Max environment is not always simple.

The Toolkit

The tools added in the new iteration of the FluCoMa toolbox (Tremblay et al. 2019) are well established in data science and most have been previously used in audio signal-processing research. Our aim is to enable programmatic interaction with sound collections and other signal corpora within CCEs to enable future research around sustained musical practice with these sorts of tools and concepts. In the toolbox's previous iteration, we pursued this aim with a focus on musical approaches to signal decomposition and description. The new additions comprise a suite of more than 30 objects available for each of the most widely used CCEs based on a common C++ architecture. Those CCEs encompass Max, SuperCollider, and Pd; a command-line interface is also under development. All code is open source, under a Berkeley Software Distribution license (BSD-3), and available on the GitHub repository of the project. Visit http://flucoma.org for the source code and the latest binaries.

The following text discusses the design for this new suite of tools and then describes its main functionality.

Aims and Priorities

Our foci for the toolkit design stem from wishing it to be useful as a general framework in its own right, as well as from broader goals of the FluCoMa project. The following aims and priorities inform our design decisions and trade-offs, and they draw on our experiences with our previous work in this area (cf. Harker and Tremblay 2012), on lessons from developing the first iteration of our toolbox, and on continuous feedback from a pool of musicians working with the tools from the earliest stages of development.

1. Native integration.
   Our tools should respect as closely as possible established idioms in the host CCE and allow easy transfer of data between the framework and native data structures.
2. Consistency.
   Objects should fit together neatly and offer a consistent way of working. We have opted for an analogue to the scikit-learn (Pedregosa et al. 2011) naming conventions for our algorithms.
3. Learnability.
   The overall framework and the granularity of its objects should afford early, easy exploration yet offer rich scope for deeper experimentation.
4. Configurability.
   Objects should offer fine adjustment and tweaking where the algorithm supports it, ideally in programmatic ways.
5. Scalability.
   Algorithms should be able to cope with reasonably large collections of data, within the confines of currently available computing power in central processing unit–based personal computing hardware.
6. Breadth.
   An initial offering of well-known and understood algorithms, touching on a breadth of possible approaches, helps us draw on existing knowledge and practice, and helps assess the kinds of extension that might be of interest to the community.
7. Completeness.
   We aim to enable complete workflows from corpus-building, curation, machine listening, and machine learning to sound making.

Additionally, because of the project's overarching aim to catalyze more and better musicking research in this area, close attention has been paid to interface consistency across different CCEs, to enable researchers working in different environments to exchange ideas and insights. Variations on similar ideas can be explored iteratively within the distinct idiomatic contexts that different CCEs afford (McPherson and Tahıroğlu 2020).

Finally (and perhaps most elusively), is a commitment to providing tools that are "artist spec" (Buxton 1997): As well as being sufficiently robust and performant for use in artistic projects, the code base and supporting resources need to be sustainable well beyond the lifetime of the research project.

Clearly some of these goals are in tension with each other; we believe, however, that the resulting toolkit strikes a reasonable balance and makes a valuable contribution to the goal of helping to animate current and future research. The following discussion briefly describes the different object categories in the framework. The following sections will outline some of the workflows these make possible.

DataSets

The DataSet object is the core component for all data-processing algorithms. It is a key-value store with unique string identifiers as keys and numerical vectors as values. The purpose is mainly to store feature vectors, but it aims to be as generic and versatile as possible within that scope. Other objects in the toolkit consume and produce DataSets as input and output.

DataSet entries are mostly introduced via the CCE's audio buffer objects, which is the main interface in our audio feature-extraction objects. Buffers are familiar to creative coders and the only open-ended memory access available in all music CCEs. Many of our objects copy data to new DataSets, rather than having many objects depend on shared DataSets. We found this approach works well for practical amounts of data using current computers and is typically much easier to reason about than shared access, especially in multithreaded environments.

DataSet instances have a JSON interface, as do all other objects in the toolkit that hold significant amounts of data and variables in their state. This allows a widely supported way to store, back up, and transfer data into different environments, as well as loading to and from native CCE dictionary structures. A sibling object is the LabelSet, which holds labels instead of feature vectors, and is used particularly in classification tasks.

Nearest Neighbors

Similarity queries are perhaps the most well-established mechanism in descriptor-driven corpus-based music making, such as concatenative synthesis or automated orchestration. Usually implemented via nearest-neighbor algorithms, they allow the association of sounds by proximity according to some set of acoustic features or derived metric. Following established practice, we implement a k-d tree to perform efficient queries. Our implementation allows querying within a given radius around a point and can report the distances to the returned points to allow intuitive use of this feature. This algorithm is often significantly better than brute-force search, although limited with respect to data dimensionality and dynamically adding or removing tree entries.

Supervised Machine Learning

Supervised machine learning models a relationship between input and output data, both of which are supplied by the user. It is well established in creative musical contexts, as has been shown in longitudinal research examining sustained practice (Fiebrink and Sonami 2020), as well as in many disciplines involved with music and audio analysis. This type of machine learning can be used both for classification (categorizing inputs into discrete classes) and for regression (devising a continuous mapping between a given set of inputs and outputs). These kinds of tasks can be quite intuitive for musicians, especially if they have experimented with machine listening and tried to handcraft code for creative purposes.

The toolbox provides both k-nearest neighbor (KNN) and multilayer perceptron (MLP) regression and classification. The KNN models are lightweight and very quick to train and to query, and they are conducive to experimentation. Still, they are limited in the complexity of relationships they can model. Models using MLP are more appropriate for problems that are more difficult: Our implementation allows flexible specification of the architecture and a choice of activation functions, which affords a great number of configurations and combinations. The neural network regressor implementation also allows both input and output access to any layer, so an MLP can be used as a feature extractor or to learn complex mappings: For example, an autoencoder

(Bourlard and Kamp 1988) setup can be used to learn hidden representations from data, and our interface allows this learned representation to be queried with new data. Although ever more possibilities and layer types keep appearing in the deep-learning community, our estimation was that a tailored implementation has better chances of providing a good balance of flexibility and complexity within the computing power available to most of the creative-coding community. Finally, MLPs are quite amenable to the relatively small sizes of musicking datasets, and even outliers or noise may still produce fruitful artistic results.

Unsupervised Machine Learning

In unsupervised learning schemes, only input data are provided, and the algorithm attempts to model the data directly. Although this means that the resulting outputs may be more abstract, this approach requires less preparation and can be advantageous for learning general features or spaces that relate items in a DataSet. Given the arduousness of obtaining supervised training data for arbitrary creative endeavors, especially in the midst of what Impett (2000) calls the "grip-slip relationship between vision and realization," we expect unsupervised machine learning to be particularly useful in data-driven musicking.

Although recent developments in general machine learning have not been as spectacular as in the supervised case, in our view there is still much promise for establishing existing tools and paradigms for unsupervised machine learning in creative domains. Our main foci are data clustering and dimensionality reduction.

With respect to data clustering, the toolbox includes an implementation of the classic algorithm for *k*-means clustering, which has a long tradition in audio signal processing (Lloyd 1982). Although the output of clustering may seem a bit abstract for a creative workflow, it can be used as a building block for complex interfaces. We plan to add other clustering algorithms to overcome the classic limitations of *k*-means, namely, globular shapes and a predefined number of clusters, as well as metrics for assessing clustering quality.

The use of dimensionality reduction is perhaps more straightforward, as it has direct applications for visualization and parameter mapping, as well as preprocessing data for other tasks in machine learning. In previous research we demonstrated the creative potential for adaptive interfaces based on the results of different dimensionality-reduction algorithms (Roma, Green and Tremblay 2019a). Based on the intuitions gained, we provide a selection in the toolset: principal components analysis (PCA), which is also generally useful for linear mapping and preprocessing of high-dimensional data; multidimensional scaling (MDS), which allows easy experimentation with different distance measures between data points; and uniform manifold approximation and projection (UMAP), which can be seen as the state-of-the-art algorithm for dimensionality reduction. Autoencoders based on MLP, discussed earlier, can also be used for nonlinear dimensionality reduction.

Data Scaling and Normalization

Data scaling and normalization are generally necessary when using any of the described machine-learning algorithms, as imbalances in the scale of the different dimensions can make it very hard to converge in multidimensional spaces, and some models expect inputs to be centered or scaled. The toolbox provides a set of common preprocessing objects for normalization, standardization, and robust scaling, alongside other housekeeping processes such as thresholding and outlier removal.

Querying

Many of the early adopters had a keen interest in manipulating the numerical data in DataSets, typically those from audio features. Such an expectation is understandable, as SQL-like interfaces have been frequently used to interact with audio corpora (e.g., Schwarz et al. 2006; Akkermans et al. 2011). The toolbox provides an object to accommodate this kind of interaction, allowing manual filtering of the rows and columns of DataSets as well as compositing before further processing.

Abstractions and Utilities

Finally, utilities are provided for accommodating common workflows and dealing with specific affordances of each CCE, including buffer processing and real-time querying operations. We provide some flexible abstractions that speed up the process of assembling a DataSet from a starting collection of audio samples and files, allowing parameters for batch analyses to be expressed concisely and abstracting away boilerplate code.

Usage Examples

We believe that the value of the proposed framework rests in large part on the affordances it yields for musicians to experiment with these technologies in environments already in their creative workflow, so as to stimulate more and better-documented practical research on creative strategies and tactics around their use in practice. A key methodological commitment of the project is that our designs are informed and interrogated at the earliest opportunity through intense use by musicians outside the development team. We were lucky to benefit from the input of a group of commissioned artists as well as a pool of enthusiastic and engaged voluntary alpha users. In the following text we will describe a few examples of creative workflows enabled by the toolbox, followed by a more detailed example from a piece by the first author, to provide an idea of how objects can be applied to musical practice. As previously stated, we hope that the resulting discussions around such practices will go beyond implementation details and engage across the various communities around each respective CCE. To encourage such an outcome, we commissioned musicians covering a range of practices, all fluent in Max or SuperCollider, to test the ability to integrate our framework into their workflows and to challenge our early interface designs.

Data-Driven Workflows

As well as incorporating the toolset into their musicking, we asked that our early coinvestigators contribute to a series of online presentations discussing their projects and experiences; these presentations are available at www.flucoma.org/plenaries. In addition, Jacob Hart, a musicology researcher on the project, has amassed a valuable archive of analysis, interviews, and code as part of his work on tracking the creative processes of our commissioned musicians.

Some common patterns of workflows have been observed in these early entanglements and can be grouped into the categories corpus building, dynamic querying, exploring parameter spaces, and adaptive interfaces.

Corpus Building

The tools for statistical processing (scaling, clustering, and dimensionality reduction) have been used to clean and filter sound corpora obtained in different ways (e.g., mass generation, sonification, or informal recording). This includes removal of outliers or near duplicates, finding good references and ranges for intuitive descriptor values, and sanitizing data across dimensions. Importantly, the interchangeability of the components of this process has allowed our early users to compare their intuition of scaling and distance with computed results in various descriptor spaces. These intuitions and the computed results often differed, which was a source, in equal parts, of bemusement and of inspiring serendipity for our artists.

Dynamic Querying

As opposed to fixed workflows based on similarity queries in concatenative synthesis or automatic orchestration tools, our toolbox's modular interface allows for an approach to complex material that is more nuanced. For example, sounds can be modeled into different segment decompositions (e.g., stages of the energy envelope), which are analyzed separately and then either merged or indexed into multiple descriptor spaces. Similarity queries can also be *forked*: Different indices can be queried depending on the value of a given descriptor. A typical example is using pitch features only for voiced segments of

Figure 1. A typical workflow for similarity matching, showing progression from source audio from steps of segmentation and analysis, statistical processing, dimensionality reduction, through to indexing. In the toolkit, there are multiple options available for these steps, indicated next to the relevant boxes. PCA = principal components analysis; MDS = multidimensional scaling; UMAP = uniform manifold approximation and projection.

a certain duration, otherwise reverting to spectral centroid.

Exploring Parameter Spaces

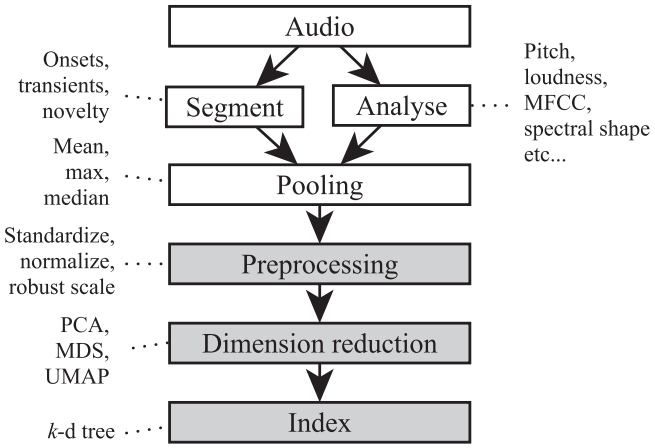
Machine learning can be used to create nonlinear mappings of parameter spaces of complex synthesizers and processors. For example, MLP objects are configured and trained to expand from a few control dimensions (which can be more easily mapped to input devices and interfaces) to many synthesis parameters. In the use experiences of our early adopters, the proposed interface allowed for ad hoc exploration of larger synthesis parameter spaces, taking advantage of small, curated training datasets. Furthermore, experimentation with regression between synthesizer parameters and the acoustic features of the resulting sounds allowed for inspiring audio-driven resynthesis models.

Adaptive Interfaces

As shown by Roma, Green, and Tremblay (2019a), interactive scatter plots for playing sound corpora can be extended using dimensionality reduction and visualization (which can also represent clusterings and classifications) to make playable interfaces that adapt to the data distribution. Simple examples include interaction with a touchscreen or other controllers, or sequencing of spatial trajectories and patterns. A more sophisticated interface developed by one of the authors allows for the use of live coding to control a number of playback heads navigating a visualization of a sound collection. This visualization is generated via dimensionality reduction, and also displays the items with descriptor-based sound icons. This interface affords a view of both structure and content at a glance. Each playback head follows a trajectory that can be recorded interactively or live coded as a function that generates a new position based on the current position.

Tuning Nearest Neighbors

Figure 1 shows an example of dynamic querying in more detail. This comes from a piece entitled



"Newsfeed," a studio composition by the first author in which similarity-based queries were used to create new utterances by concatenating various segments from a corpus of spoken voice material.

For context, Figure 1 shows a typical workflow for similarity matching, alongside options available in our toolkit for each stage. For training, each stage will normally be performed in one batch. Collections of host buffers are converted to a DataSet, and each modeling stage produces a new DataSet that serves as input to the next stage. At query time, new data points (in CCE buffers) are passed through the pipeline to the *k*-d tree, which returns the nearest sounds from the corpus.

In practice, a great deal of experimentation is required. The granularity of segmentation, choice of features, pooling mechanism, and so forth, can significantly affect the behavior of the similarity query at the end of chain. Moreover, in a creative context, the desired musical behavior is often neither known nor fixed, instead it is negotiated through an interactive process of programmatic musicking, in which exploring options and their impact on sonic results in context transforms musical questions in a fully circular process.

The first author found that he was able to enrich the workflow above with custom tools to fluidly and interactively explore options during the composition process. The ability to filter and merge different DataSets using the DataSetQuery object meant it was possible to quickly try out

different combinations of descriptors, scaling schemes, and dimension-reduction models. Additionally, the segmentation could be adjusted ex post facto by inserting a clustering model into the chain prior to pooling: Starting with a deliberately fine-grained segmentation, the *k*-means method was then used to identify adjunct segments that could be rejoined according to selected descriptors, enabling an effective way to discover various time scales and groupings that worked well in different contexts.

Crucially, new ideas and possibilities suggested themselves during these explorations in a way that would have been much less likely in a setting that was less interactive and more disjointed, as would have been the case with a solution that was less configurable. Moreover, this creative hacking workflow was able to integrate well into the first author's distinctive practice, which bridges working in CCEs and digital audio workstations in ad hoc ways, at times through audio sends, at other times via MIDI events, or even full edit decision lists. These were easily thrown together in context to best fit the need of the musicking moment.

This last example, as well as a few others from the pool of early users, are available both in the online presentations mentioned earlier in this article and in code snippets on the project's online forum at discourse.flucoma.org.

Preliminary Assessments

The first version of the new extensions was released to our community of creative coders in November 2019. At that point the community comprised eleven participants plus the authors, augmented by a small group of eager people along the way. Throughout the process, we released nine private updates and then transitioned to a public beta phase at the end of April 2021. Although it is still too soon to reflect in detail on how well the proposed framework supports sustained musical research, and notwithstanding the encouraging range of possible uses described above, we have noted some early trends in people's encounters with the tools that will guide our future work.

Overall Framework Aims

The variety and sophistication of the workflows described in the previous section provide encouraging signals that our aims for the toolkit are being fulfilled. Our early adopters have been able to integrate the tools into their diverse range of broader working patterns and to complete a variety of new work that would have been otherwise much more difficult. This suggests that the tools themselves are sufficiently robust, performant, and usable to support serious creative work, and that the coverage of algorithm types provides enough breadth for a range of ideas to be explored and brought to fruition. From a framework development perspective, the C++ architecture has made it easy to add and adapt objects to the toolbox.

At the time of writing, some avenues for enhancement are already clear, based on feedback from our contributing artists and the authors' own experiences of creative work with the tools. Most prominent among these are some rough edges around idiomatic integration into CCEs vis-à-vis the desired ergonomics of our higher-level vision of fluid corpus manipulation. Indeed, the goal of ensuring the consistency of our interfaces across CCEs can often be in tension with the goal of enabling familiar and idiomatic usage in a particular environment. An area for some further development, now that there is a common basis in place, is to look at potential improvements in environment-specific integration.

This is most apparent in the way in which we have used audio buffers as a type of scalable and configurable container that has an equivalent available in all our target environments. Although this approach has enabled rapid and simultaneous development for these diverse architectures, as well as providing fast, real-time access to scalable data structures, it has also yielded interfaces that can be slightly cumbersome in all environments. A possible approach to improving this could be to apply idioms that are both higher level and more familiar on top of these lower-level (but consistent) mechanisms.

Now that a certain amount of usage experience has accumulated, it is easy to see what these idioms might look like, given the common types of "boilerplate" code that we and our collaborators

have had to repeatedly produce. For instance, in Max and Pd, we could provide a mechanism for dealing transparently with CCE-native lists to store and recall data in containers and to trigger algorithms. Likewise, on the SuperCollider server, an equivalent abstraction would afford working directly with control-rate signals without needing to address and access buffers.

Wider Project Aims: Human Learning

Besides some possible areas for ergonomic improvement and functional expansion discussed previously, it is worth highlighting some areas of conceptual difficulty that early adopters have encountered with these new tools, not least as these can be suggestive of ways in which specifically musical applications of such technologies may depart from more established usage.

The workflows presented in the previous section took shape in a context of continual exchange and discussion, on the project forum and through regular online group meetings in which early adopters shared work in progress on their respective projects. Such communication on musical mining has been helped by the consistency of concepts and interfaces between the Max and SuperCollider implementations. This communal approach has been significant not just in showing what can be done, but in helping to form shared understandings and research ideas, particularly across different disciplines and styles. It also indicated areas in which the broader project could help musicians become fluent with, and critical of, these technologies.

Just as importantly, this group of techno-fluent composers was able to provide feedback on the learning curve of working with concepts from data science while using our emerging toolbox. We have noticed, in particular, that for those with little prior experience in the language and techniques of data science, the conceptual jumps can seem quite daunting: People are confronted with abstract features and procedures far removed from musical intuitions, and with the task of relating objective validation to subjective expectations. Moreover, the general hyperbole surrounding the potential of machine learning and machine listening requires a careful and tactful calibration of expectations while still showcasing the creative affordances they provide when these technologies are embraced divergently.

By giving our early adopters a forum in which to voice their concerns, they were able to challenge the need to understand some of the more obscure data science concepts and their usefulness in actual musicking. This approach, through frank dialogue, was useful to understand where and when to stop with technical explanations and when to use metaphors.

Broadly speaking, many of these episodes have a common theme. The objects in our second offering represent a jump in abstractness from those introduced first. We departed from algorithms that already had a basis in musical signal processing and analysis, and we delved into a range of algorithms more simply concerned with data in the general sense. In the following we spell out the most salient of these difficulties.

Dimensionality

Thinking beyond three dimensions is hard and quickly becomes highly abstract. Becoming comfortable with this mental exercise is, in part, simply a matter of practice. Many algorithms start to perform markedly worse as the dimensionality of input data increases, but the extent to which this palpably degrades the results obtained depends not only on the algorithm but also on the data and what one is trying to achieve. This represents a pedagogical challenge in giving sufficiently useful rules of thumb to observe, in terms of which models might work reasonably given some quantity of multidimensional input data. In particular, what counts as "a lot" of dimensions can vary markedly between different algorithms, as can the interaction between this dimensionality and the quantity of training examples.

Breakdowns in terms of dimensionality are especially acute when dealing with models that use distance metrics between multidimensional input points. When trying to develop models in which a given algorithm's notion of how similarly two corpus items correlate with a human's perspective,

it is reasonable to, at first, expect that adding more input features might improve this correlation by giving the computer more data to work with. Although as human listeners we are able to latch fluidly and rapidly onto different saliences, the "curse of dimensionality" in machine learning means that many algorithms simply struggle as the number of input dimensions increases.

Data Quality

The dependence of models on the quality of their training inputs is a well-known problem in machine-learning applications, but not one that admits of very simple explanations. It is not trivial to ensure that those factors that could affect the notional quality of training data in a musicking context are understood. Such factors could include the abundance of data, the consistency of labeling, the degree to which the data's dimensions are correlated, how the data are statistically distributed, how much coverage the training examples offer of expected future inputs, the relative scales between dimensions, and how input scales relate to perceptual correlates.

In practical terms, the codependence between data, algorithm, and desired outcomes means that coming to terms with each of these points requires iterated experimentation for each new application. Although such empiricism is not alien to musicians, the number of coupled factors and their relative abstractness can lead to confusion or frustration. For instance, to improve results, one might try adjusting the "hyperparameters" of an algorithm, whereas it could often be better (but more elusive) to try "improving" the input data. Although the toolset provides the means to subjectively assess model performance in relation to musical goals, we have found in practice that the temptation to keep tuning models in preference to such "sanity checks" can be strong and can sometimes lead musicians astray.

This often-frustrating tinkering of various codependent moving parts is especially acute in the case of the MLP, and neural networks in general, where finding a workable arrangement of network architecture and hyperparameters for given training data is necessarily a case of trial and error. As such, a clear avenue for further research in this area is to arrive at additional ways of assessing input data, evaluating model results, and developing useful guidelines for improvement, such as when to add more or different data, when to preprocess, and so forth.

Units and Scales

When working with the data analysis objects, one quite quickly ceases to have data in physically relatable units like MIDI pitch or dB—however perceptually approximate—and instead one needs to contend with data whose units and scales are more abstract. Although this is not in and of itself a problem for creative coding musicians, who deal routinely with dimensionless ranges (for instance, when mapping between controllers and processors), developing an intuition for what these new quantities might represent can be challenging.

This is particularly true of dimensionality-reduction processes, in which the nature of the outputs depends on the inputs in a way that can be hard to anticipate. For instance, even for the simplest such process in our toolkit, PCA, the mapping it performs is not easy to explain in general physical terms, and coming to terms with the range of the output data requires some familiarity with the concept of statistical variance. This has caused particular problems, in our experience, when artists have wanted to relate the products of independent dimensionality reductions to each other. In general, there is no simple mechanism to do this, because each result is derived wholly from the input data the model was shown. But understanding why, or how to get around it, requires an engagement with the implementation details of an algorithm that may be daunting.

Future Work

At the time of writing, a small community of users is gathering momentum, but a few clear next steps are emerging, both in terms of technological affordances that are currently missing in the toolset, and in terms of our ambitious aim to foster a wider diverse community of critical creative coders.

Potential Framework Extensions

Early experiences with the framework have suggested some possible additions to the toolkit for us to consider. Concentrating on first providing well-established and well-understood algorithms has enabled swift development, as well as taking advantage of a range of learning resources online and in the literature. We anticipated that by seeing what people did with these resources and noting where blockages occurred, a principled way to draw on more recent developments or extend existing techniques would emerge. Some examples of this happened quite quickly and have been written about in other publications—for instance, implementing the comparatively recent UMAP algorithm to supplement other, more-established, dimensionality-reduction objects, as well as proposing a range of musically targeted applications of existing techniques such as nonnegative matrix factorization (NMF; discussed by Roma, Green, and Tremblay 2019b; for an introduction to the NMF technique itself, cf. Lee and Seung 1999) and using similarity graphs (Roma, Tremblay, and Green 2021).

In the final year of the FluCoMa project, we intend to continue growing the toolbox with new objects and new object categories. Several signal-processing tools for "hybridizing" sounds and interpolating gestures are already available (Roma, Green and Tremblay 2020). Moreover, we have postponed further optimization of the code to this last segment: By observing the toolkit's performance in practice, and what sorts of size and complexity of collection people end up gravitating towards, we hope to streamline this process with a particular perspective on more investigation of the real-time possibilities and hurdles created by musician use.

Meanwhile, based on our own creative work and feedback from our collaborators, some possible avenues for expansion can be described as: temporally aware analysis; dynamic, high-dimensional indexing; alternative time-frequency representations; and validation tools.

Temporally Aware Analysis

Time is, of course, of foundational importance to sonic and musical creative work, and many of the algorithms we currently provide do not handle it explicitly, which can lead both to confusion and to extra effort on the part of users. One avenue would be to look at making some algorithms available that are geared at analyzing the temporal evolution of features. There is a range of techniques that could be explored, such as convolutional extensions to NMF (Smaragdis, Raj, and Shashanka 2008), or the various approaches to recurrent neural networks from reservoir computing (Kiefer 2014, 2019).

Dynamic, High-Dimensional Indexing

Although *k*-d trees are easy to implement and relatively lightweight, they suffer from being costly to modify after construction and from scaling poorly to higher dimensionalities. One possible addition would be to explore approximate nearest-neighbor algorithms (Slaney and Casey 2008), which scale more gracefully (at the expense of exact matching), as well as alternative, mutable tree structures that would allow corpora to be more easily constructed and queried in real-time.

Alternative Time-Frequency Representations

Extensions of and alternatives to the well-established Fourier-domain analysis of audio, such as constant-Q transforms (CQTs), the scattering transform (Salamon and Bello 2015), or auditory models (Lyon et al. 2010), are by no means new, but remain broadly unexplored in CCEs. Possible reasons for this may be the lack of invertibility of many CQT implementations until recently (Necciari et al. 2013), as well as a lack of experience of what modifications or analyses are practical and useful in the transform domain. Our framework is, we believe, flexible enough to provide a basis for musicians to start exploring some of these questions themselves, should implementations be made available.

Validation Tools

One final area of further work would be to establish a range of tools for assessing learned models in supervised-learning workflows, to help streamline their use.

Breadth of Audience

Our target audience so far has been "fluent" users of CCEs: See Green, Tremblay, and Roma (2018) for a discussion of how we conceptualize ideas of technological dispositions in relation to techno-fluency. In this way we have hoped that usage experience will help catalyze a virtuous circle in which multiple "means of entry" can be curated to afford less-fluent or more-casual engagement, which in turn would feed new, divergent, inclusive ideas into the community. This is of especial importance to the suite of data-oriented objects discussed here, in which the extra degrees of abstraction that come with machine-learning processes that are more generic may not be immediately musically suggestive to all. A number of tactics may well be fruitful here, discussed in the rest of this section.

Online Resources

Work is currently underway on developing a platform for the support of learning resources that respond to the kinds of need identified in the previous section, as well as clarifying and supplementing not only the documentation and examples available with the current package but also the knowledge base accumulating on our forum. In addition, we will run workshops and publish workshop materials and essays, with the hope that we continue to benefit and learn from diverse feedback and discussions that can inform future technical and critical work in this area. The goal here is to target a wide breadth of familiarity and to retain focus specifically on working with these technologies in a musical context, while pointing to more general material for those who wish to go "deeper." Again, granularity of information and interface to knowledge and applications is our main research focus, and we hope to be able to tap into our first series of workshops to enable as many inclusive entry points as possible for the complex questions raised by machine listening and machine learning.

Enabling Code Snippets

The accumulated experience from our early releases, commissions, and first workshops reveals a few recurring tasks that can be encapsulated in the distributed package. For example, exploring a corpus visually in a 2-D space, as popularized by CataRT (Schwarz et al. 2006) and continued by AudioStellar (Garber, Ciccola, and Amusategui 2021), provides a useful abstraction in its own right, as well as a fruitful way to explore the practical effects of different dimensionality reduction strategies. A careful consideration on how much a specific workflow is implied in such snippets is always at the forefront of their design.

Higher-Level Idioms

Encapsulation can be taken further by exploring different idioms altogether. For instance, we could follow the example of the Vizzie package for Max (see http://cycling74.com/articles/introducing-vizzie) as an alternative paradigm for working with Jitter for video processing; we could develop Max4Live devices aimed specifically at common workflows with Ableton Live; or we could develop wrappers for additional host environments, such as browsers, via a cross-compiler like Emscripten or game engines like Unity3D.

One motivation for "higher-level" interfaces is that a set of options that is more curated (and so necessarily more restricted) can, in practice, be more attractive or more empowering to a range of potential contributors. The danger here, however, is that we simply entrench too many of our ideas of musical workflow into a less flexible setting.

Our long-term hope is that engagement from a broad cross-section of the creative coding music community will help to turn this into a collaborative effort that supports and informs sustained, collaborative musicking research.

Critical Reflection on Method

Early in the project (cf. Green, Tremblay, and Roma 2018) we situated our methodological approach in the context of critical philosophy of technology (Feenberg 2017) and recent critical work on interdisciplinarity (Barry and Born 2013), with the aim that this would not only help strengthen our particular

musical and technical goals with this project but also form a basis, more generally, for combining research into artistic practice with research into technical design. As such, we have hoped to work towards what Barry and Born have called a "logic of ontology," in which interdisciplinary work leads to a shared shift in understanding of the objects of study, in contrast to situations where disciplines merely end up in service to each other.

These aspirations warrant some sustained interrogation, beyond the scope of this article, as we approach the end of this first extensive period of research. Work on this basis is proceeding, concerned with examining how our aspirations for the toolkit's ecosystem and the cross-disciplinary methodology that produced it fared in practice, and what lessons we might draw upon to inform future efforts at music technology research that emphasizes accountability to diverse artistic communities.

Conclusions

This article presented the second iteration of the Fluid Corpus Manipulation toolbox, a new software framework for programmatic mining of sound banks in CCEs. Combined with its previous iteration, it offers a broad system that enables programmatic data-driven musicking for interaction with audio and other data corpora within popular creative coding environments. Early usage and feedback from a community of users suggest that, through several iterations, the toolbox has fulfilled its design goals with respect to native integration, consistency, learnability and configurability of interface, scalability, and breadth, while enabling complete mining-as-musicking workflows. We have shown a range of possibilities that emerged from commissioned works, and we hope that further interest in the toolkit will be stimulated by the premieres of these works, as well as their documentation. We also hope that by reporting the various learning hurdles needed to enable an empowered fluency with these technologies, and by reflecting on our next challenges and on the potential additions to the toolkit, we will foster an open, inclusive, and critical research community spanning coding environments, institutional borders, and disciplinary enclaves.

Acknowledgments

This article is an extended version of our paper presented at the International Computer Music Conference (Tremblay, Roma, and Green 2021). We would like to thank the creative coders engaging in the alpha community (James Bradbury, Rodrigo Constanzo, Richard Devine, Alice Eldridge, Daniele Ghisi, "Leafcutter" John Burton, Lauren Hayes, Ted Moore, Olivier Pasquet, Sam Pluta, Hans Tutschku), the CeReNeM and its Creative Coding Lab, and the European Research Council, since this research was made possible thanks to a project funded under the European Union's Horizon 2020 Research and Innovation Program (grant agreement no. 725,899).

References

Agostini, A., and D. Ghisi. 2012. "Bach: An Environment for Computer-Aided Composition in Max." In *Proceedings of the International Computer Music Conference*, pp. 373–378.

Akkermans, V., et al. 2011. "Freesound 2: An Improved Platform for Sharing Audio Clips." In *Proceedings of the International Conference on Music Information Retrieval*, pp. 3–5.

Barry, A., and G. Born. 2013. "Introduction." In A. Barry and G. Born, eds. *Interdisciplinarity: Reconfigurations of the Social and Natural Sciences.* London: Routledge, pp. 1–56.

Bourlard, H., and Y. Kamp. 1988. "Auto-Association by Multilayer Perceptrons and Singular Value Decomposition." *Biological Cybernetics* 59(4–5):291–294. 10.1007/BF00332918, PubMed: 3196773

Buitinck, L., et al. 2013. "API Design for Machine Learning Software: Experiences from the scikit-learn Project." In *European Conference on Machine Learning and Principles and Practices of Knowledge Discovery in Databases*, pp. 108–122.

Bullock, J., and A. Momeni. 2015. "ml.lib: Robust, Cross-Platform, Open-Source Machine Learning for Max and Pure Data." In *International Conference on New Interface for Musical Expression*, pp. 265–270.

Buxton, B. 1997. "Artists and the Art of the Luthier." *ACM SIGGRAPH Computer Graphics* 31(1):10–11. 10.1145/248307.248315

Carpentier, G., et al. 2012. "Automatic Orchestration in Practice." *Computer Music Journal* 36(3):24–42. 10.1162/COMJ_a_00136

Collins, N. 2011. "SCMIR: A SuperCollider Music Information Retrieval Library." In *Proceedings of the International Computer Music Conference*, pp. 499–502.

Feenberg, A. 2017. *Technosystem: The Social Life of Reason*. Cambridge, Massachusetts: Harvard University Press.

Fiebrink, R., and P.R. Cook. 2010. "The Wekinator: A System for Real-Time, Interactive Machine Learning in Music." In *Proceedings of the International Conference on Music Information Retrieval*. Available online at archives.ismir.net/ismir2010/latebreaking/000012.pdf. Accessed January 2022.

Fiebrink, R., and L. Sonami. 2020. "Reflections on Eight Years of Instrument Creation with Machine Learning." In *International Conference on New Interfaces for Musical Expression*, pp. 237–242.

Garber, L., T. Ciccola, and J. C. Amusategui. 2021. "AudioStellar, an Open Source Corpus-Based Musical Instrument for Latent Sound Structure Discovery and Sonic Experimentation." In *Proceedings of the International Computer Music Conference*, pp. 62–67.

Ghisi, D., and C. Agon. 2016. "Real-Time Corpus-Based Concatenative Synthesis for Symbolic Notation." In *Proceedings of the International Conference on Technologies for Music Notation and Representation*, pp. 7–13.

Green, O., P. A. Tremblay, and G. Roma. 2018. "Interdisciplinary Research as Musical Experimentation: A Case Study in Musicianly Approaches to Sound Corpora." In *Proceedings of the Electroacoustic Music Studies Network Conference*. Available online at www.ems-network.org/spip.php?article471. Accessed January 2022.

Hackbarth, B., N. Schnell, and D. Schwarz. 2010. "AudioGuide: A Framework for Creative Exploration of Concatenative Sound Synthesis." Research report. Paris: IRCAM. Available online at articles.ircam.fr/textes/Hackbarth10a/index.pdf. Accessed January 2022.

Harker, A., and P. A. Tremblay. 2012. "The HISSTools Impulse Response Toolbox: Convolution for the Masses." In *Proceedings of the International Computer Music Conference*, pp. 148–155.

Impett, J. 2000. "Situating the Invention in Interactive Music." *Organised Sound* 5(1):27–34. 10.1017/S1355771800001059

Kiefer, C. 2014. "Musical Instrument Mapping Design with Echo State Networks." In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pp. 293–298.

Kiefer, C. 2019. "Sample-level Sound Synthesis with Recurrent Neural Networks and Conceptors." In *PeerJ Computer Science* 5:Art. e205. 10.7717/peerj-cs.205, PubMed: 33816858

Lee, D. D., and H. S. Seung. 1999. "Learning the Parts of Objects by Non-Negative Matrix Factorization." *Nature* 401:788–791. 10.1038/44565, PubMed: 10548103

Lloyd, S. 1982. "Least Squares Quantization in PCM." *IEEE Transactions on Information Theory* 28(2):129–137. 10.1109/TIT.1982.1056489

Lyon, R. F., et al. 2010. "Sound Retrieval and Ranking Using Sparse Auditory Representations." *Neural Computation* 22(9):2390–2416. 10.1162/NECO_a_00011, PubMed: 20569181

McPherson, A., and K. Tahıroğlu. 2020. "Idiomatic Patterns and Aesthetic Influence in Computer Music Languages." *Organised Sound* 25(1):53–63. 10.1017/S1355771819000463

Necciari, T., et al. 2013. "The ERBlet Transform: An Auditory-Based Time–Frequency Representation with Perfect Reconstruction." In *IEEE International Conference on Acoustics, Speech and Signal Processing,* pp. 498–502.

Pedregosa, F., et al. 2011. "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research* 12:2825–2830.

Roma, G., O. Green, and P. A. Tremblay. 2019a. "Adaptive Mapping of Sound Collections for Data-Driven Musical Interfaces." In *International Conference on New Interface for Musical Expression*, pp. 313–318.

Roma, G., O. Green, and P. A. Tremblay. 2019b. "Time Scale Modification of Audio Using Non-Negative Matrix Factorization." In *Proceedings of the International Conference on Digital Audio Effects,* pp. 213–218.

Roma, G., O. Green, and P. A. Tremblay. 2020. "Audio Morphing Using Matrix Decomposition and Optimal Transport." In *Proceedings of the International Conference on Digital Audio Effects*, pp. 147–154.

Roma, G., P. A. Tremblay, and O. Green. 2021. "Graph-Based Audio Looping and Granulation." In *Proceedings of the International Conference on Digital Audio Effects*, pp. 253–259.

Salamon, J., and J. P. Bello. 2015. "Unsupervised Feature Learning for Urban Sound Classification." In *IEEE*

International Conference on Acoustics, Speech and Signal Processing, pp. 171–175.
Schnell, N., et al. 2009. "Mubu and Friends: Assembling Tools for Content Based Real-Time Interactive Audio Processing in Max/MSP." In *Proceedings of the International Computer Music Conference*, pp. 423–426.
Schnell, N., et al. 2017. "PiPo, a Plugin Interface for Afferent Data Stream Processing Modules." In *Proceedings of the International Symposium on Music Information Retrieval*, pp. 361–367.
Schwarz, D., et al. 2006. "Real-Time Corpus-Based Concatenative Synthesis with CataRT." In *Proceedings of the International Conference on Digital Audio Effects*, pp. 279–282.
Slaney, M., and M. Casey. 2008. "Locality-Sensitive Hashing for Finding Nearest Neighbors [Lecture Notes]." *IEEE Signal Processing Magazine* 25(2):128–131. 10.1109/MSP.2007.914237
Small, C. 1998. *Musicking: The Meanings of Performing and Listening*. Middletown, Connecticut: Wesleyan University Press.
Smaragdis, P., B. Raj, and M. Shashanka. 2008. "Shift-Invariant Probabilistic Latent Component Analysis." In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 2069–2072.
Smith, B. D., and W.S. Deal. 2014. "ml.∗: Machine Learning Library as a Musical Partner in the Computer-Acoustic Composition Flight." In *Proceedings of the International Computer Music Conference*, pp. 1285–1289.
Tremblay, P. A., et al. 2019. "From Collections to Corpora: Exploring Sounds through Fluid Decomposition." In *Proceedings of the International Computer Music Conference*, pp. 223–228.
Tremblay, P. A., G. Roma, and O. Green. 2021. "Digging it: Programmatic Data Mining as Musicking." In *Proceedings of the International Computer Music Conference*, pp. 295–300.