spikely Documentation

Release 0.0.1

Roger Hurwitz, Cole Hurwitz

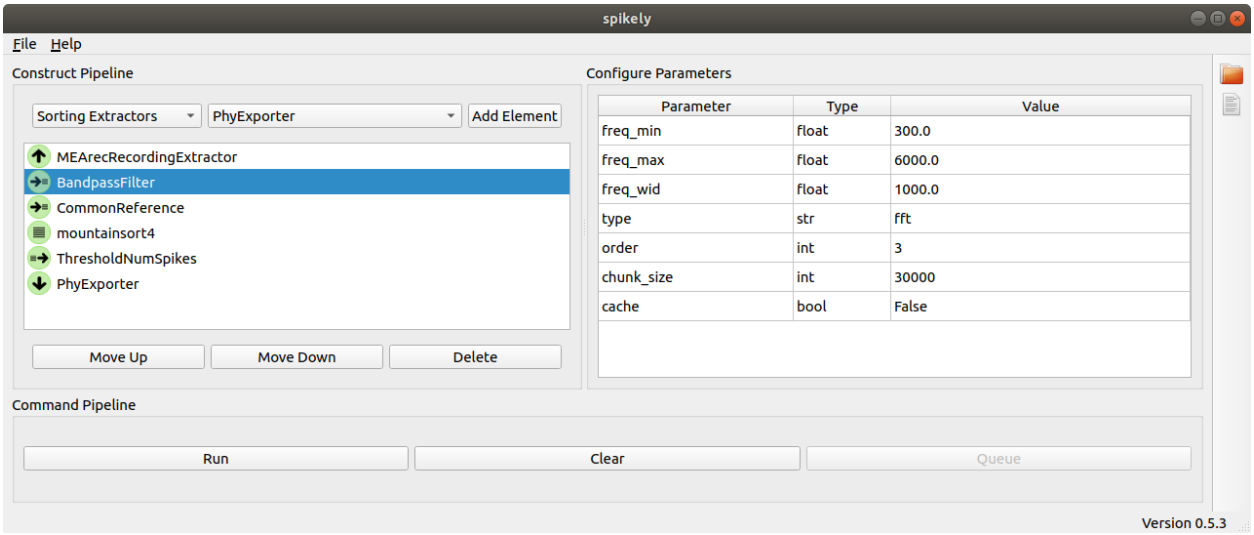
Jun 16, 2020

Contents

CHAPTER 1

Spike Sorting Made Simple

Spikely is a Python application built on top of SpikeInterface designed to simplify the process of creating and running spike sorting pipelines. Spikely supports loading, preprocessing, sorting, curating, and exporting of extracellular datasets that are stored in SpikeInterface compatible file formats.



CHAPTER 2

Contents

2.1 Overview

SpikeInterface is a powerful Python-based extracellular data processing framework supporting a broad range of features and functions. Its power and breadth come at a price, however, and that price is complexity. But for those well-versed in Python programming and needing full control over the extracellular data processing process, working directly with SpikeInterface is the way to go.

Spikely, on the other hand, is for users who want to take advantage of SpikeInterface without having to program in Python. Spikely provides a GUI on top of SpikeInterface optimized for a specific use case: pipelining extracelluar data from a source to a sink while enabling one or more data transformations along the way. In exchange for its ease of use and efficiency, spikely sacrifices some of SpikeInterface's power and breadth. Spikely therefore complements, rather than replaces, SpikeInterface.

Tip: One of scenarios we had in mind when making spikely was enabling bulk extracelluar data processing in the lab. After data collection, running a standard series of transformations may be the next step prior to a deeper dive on the data. In that case using spikely may improve overall throughput.

Because of the close relationship between spikely and SpikeInterface, it is important for the spikely user to have a grounding in the concepts behind SpikeInterface. If you are not already familiar with SpikeInterface, a good place to get started is its online documentation

In addition to being familiar with SpikeInterface, taking full advantage of spikely requires an understanding of a few key concepts specific to it:

- **Element** - An element in Spikely corresponds to entities exposed by the data processing nodes in SpikeInterface. To be used in spikely, the underlying SpikeInterface entity must already be installed on the user's system. For information on installing SpikeInterface entities like Spike Sorters, check out this document.

  Elements in spikely consist of:

  - *Extractors* - Extractors read raw extracelluar data from files, and make those data available to downstream elements in the pipeline. Extractor names correspond to the raw extracellular data format they support. Spikely requires one, and only one, Recording Extractor per pipeline.

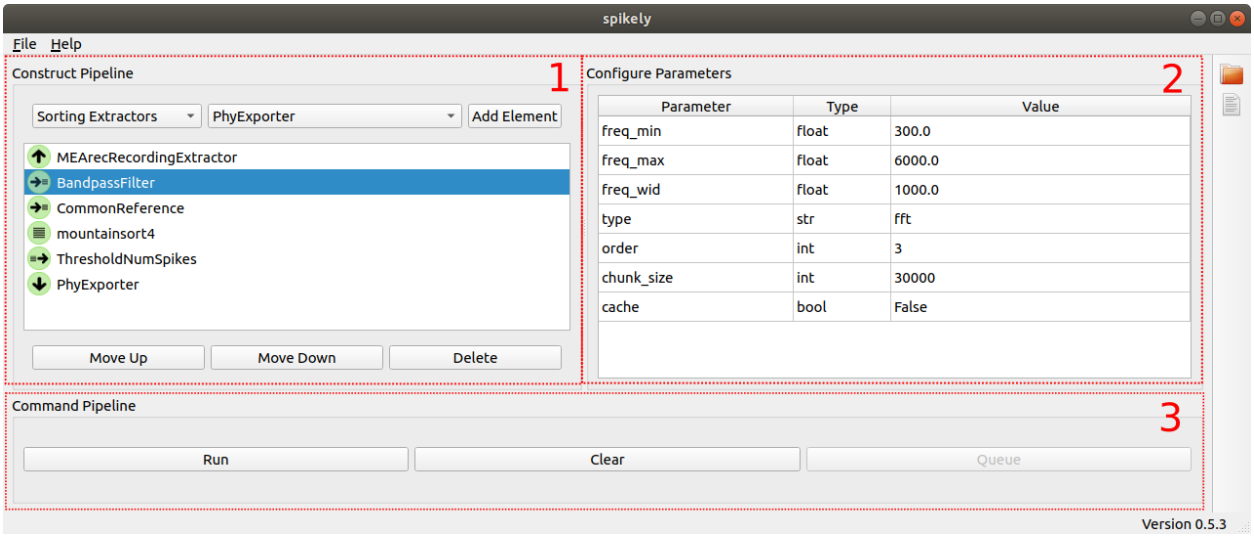
  - *Pre-Processors* - Pre-Processors transform data sourced into the pipeline by the Extractor before it is sent to the Sorter. Pre-processors are optional. Spikely supports multiple Pre-Preprocessors per pipeline between the Extractor and the Sorter.
  - *Sorters* - Spike sorting is a big part of SpikeInterface, and spikely's Sorters correspond closely to spike sorters in SpikeInterface. Spikely requires the presence of one, and only one, Sorter in the pipeline. Sorters write their results out to a file (unless specified not to) allowing a Sorter to act as a terminating sink in a spikely pipeline.
  - *Curators* - Curators, also known as post-processors, automatically curate sorted data produced by the Sorter and output them downstream to either another Curator or to a pipeline terminating Exporter. Curators are optional. Spikely supports multiple Curators per pipeline.
  - *Exporters* - Exporters act as data sinks, transforming sorted datasets into different formats. Exporters are optional, and spikely only supports a single Exporter per pipeline.

- **Parameter** - Most elements have one or more parameters associated with them that can be edited by the user in spikely to customize the behavior of that element during the execution of a pipeline. Parameters are element specific, and some familiarity with the proxied node in SpikeInterface is required to correctly configure an element.
- **Pipeline** - The user organizes elements in spikely in a series where extracelluar data "flows" from the first element in the Pipeline to the last when the pipeline is run. Pipelines, and their associated parameterized elements, can be saved for future use thereby enabling greater efficiency and repeatability.

2.2 Workflow

With a grounding in SpikeInterface, and a grasp of spikely's element, parameter, and pipeline abstractions, the last piece of the puzzle to unlocking spikely's potential is understanding its workflow and associated user interface. Spikely was designed for specific use cases, and its workflow is optimized with those use cases in mind.

At a high level spikely's workflow consists of creating a pipeline of elements, configuring the parameters associated with those elements, and finally, running the pipeline to pull extracellular data into the pipeline transforming it as it flows through to the end.



1. **Constructing the Pipeline** - The user constructs a pipeline in spikely by choosing the element category (e.g., *Extractors*), choosing one of the installed elements within that category (e.g., *MdaRecordingExtractor*) and then adding that element to the pipeline using the "Add Element" button. Individual elements added to the pipeline

can be moved up, moved down, or deleted as part of pipeline construction process. Note, there are pipeline policies enforced by spikely related to ordering and singularity that limit certain pipeline permutations.

2. **Configuring Element Parameters** - When an element is selected in the *Construct Pipeline* part of the UI that element's parameters are displayed in the *Configure Elements* part of the UI. Element parameters are specific to it, so a detailed explanation of an element's parameters will need to be gleaned from the corresponding SpikeInterface documentation. Clicking on the *Value* field for a parameter enables the user to edit it. Spikely does rudimentary type checking, but for the most part it is up to the user to ensure that a parameter value is valid.
3. **Commanding the Pipeline** - While the commands available to the user in the *Construct Pipeline' part of the UI operate on individual elements in the pipeline, *Command Pipeline* commands act on the pipeline as a whole. Currently, two operations are supported: *Run*, and *Clear*. *Clear* deletes all the elements in the pipeline enabling the user to quickly tear down the current pipeline before building up a new one. *Run* is where the magic happens, instantiating the pipeline and transforming the extracellular data as it flows from the source element (Extractor) to the sink element (Sorter or Exporter).

Tip: The pipeline creation and parameter configuration steps can be shortened by saving and loading complete pipelines to/from files using the corresponding actions from spikely's *File Menu.*

2.3 Installation

`spikely` is a Python package. The latest production version can be installed this way:

```
pip install spikely
```

If you want to work directly off the master branch of the repository, the latest pre-production version can be installed this way:

```
git clone https://github.com/SpikeInterface/spikely
pip install -e spikely
```

2.4 Contact Us

Below are the authors of spikely:

- Roger Hurwitz [1]
- Cole Hurwitz [2]
- Shawn Guo [3]

For any inquiries, please email rogerhurwitz@gmail.com or just leave an issue!

1. Independent Developer, Portland, Oregon, USA
2. PhD Candidate, The Institute for Adaptive and Neural Computation (ANC), University of Edinburgh, Edinburgh, Scotland.
3. Research Assistant, The Institute for Adaptive and Neural Computation (ANC), University of Edinburgh, Edinburgh, Scotland.