spikely Documentation

Release 0.0.1

Roger Hurwitz, Cole Hurwitz

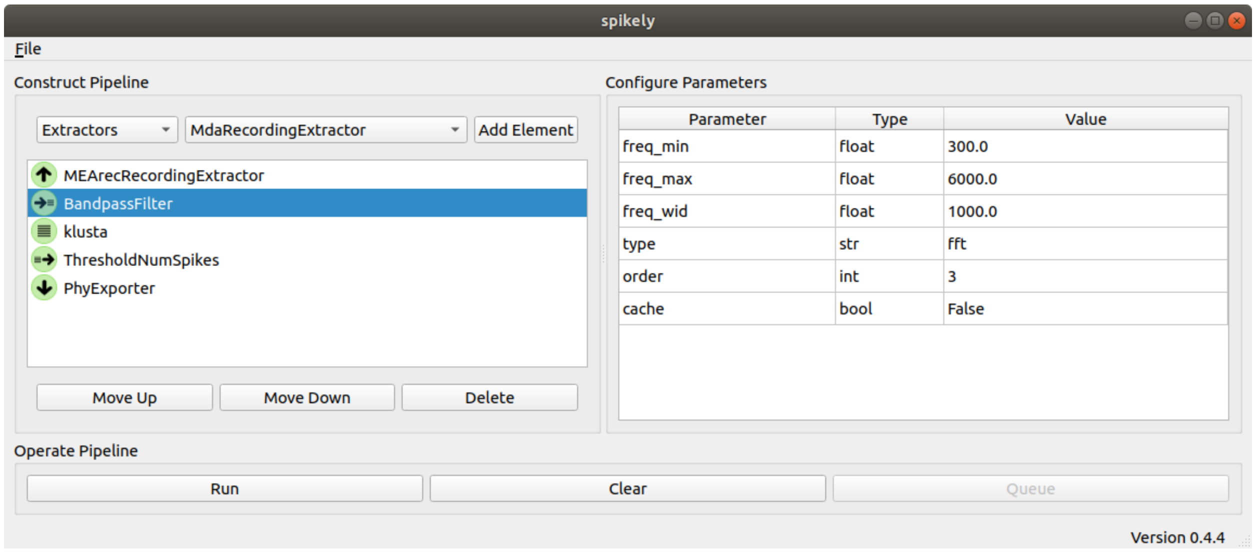
Jun 05, 2020

Contents

CHAPTER 1

Spike sorting made simple

Spikely is an 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.



Extractors | MdaRecordingExtractor

MEArecRecordingExtractor
BandpassFilter
klusta
ThresholdNumSpikes
PhyExporter

Parameter	Type	Value
freq_min	float	300.0
freq_max	float	6000.0
freq_wid	float	1000.0
type	str	fft
order	int	3
cache	bool	False

CHAPTER 2

Contents

2.1 Overview

SpikeInterface is a powerful Python-based extracellular data processing framework supporting a broad range of features and functions. 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 a subset SpikeInterface's processing power without having to program in Python. Instead, 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 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 capabilites exposed by the data processing nodes in SpikeInterface. Specifically, spikely elements 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, 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 betwween 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, and transform sorted datasets to a formatted output file. 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 therby enabling greater efficiency and repeatability.

2.2 Installation

`spikely` is a Python package. It can be installed using pip:

```
pip install spikely
```

If you want to install from the source so that you are up-to-date with the latest development, you can install with:

```
git clone https://github.com/SpikeInterface/spikely
cd spikely
python setup.py install (or develop)
```

2.2.1 Requirements

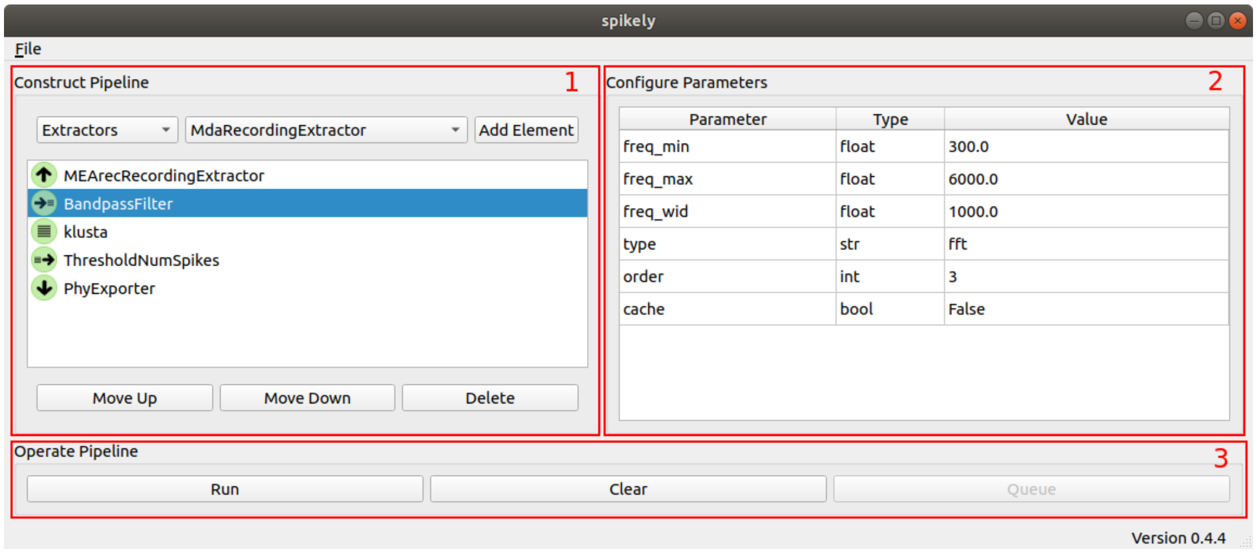
The following Python packages are required for running spikely. They are installed when using the pip installer for `spikeinterface`.

- PyQt5
- spikeextractors
- spiketoolkit
- spikesorters

2.3 Workflow

With a solid 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 UI layout.

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 cause extracellular data to be brought into the pipeline by the Extractor and transformed by the other elements in the pipeline flow.



Parameter	Type	Value
freq_min	float	300.0
freq_max	float	6000.0
freq_wid	float	1000.0
type	str	fft
order	int	3
cache	bool	False

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. **Operating the Pipeline** - While the commands available to the user in the *Construct Pipeline' part of the UI operate on individual elements in the pipeline, *Operate 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 shortcut by saving and loading complete pipelines to/from files using the corresponding actions from spikely's *File Menu.*

2.4 Contact Us

Below are the authors of spikely:

- Roger Hurwitz [1]
- Cole Hurwitz [2]

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

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