neurotic Documentation

Release 0.7.0

Jeffrey Gill

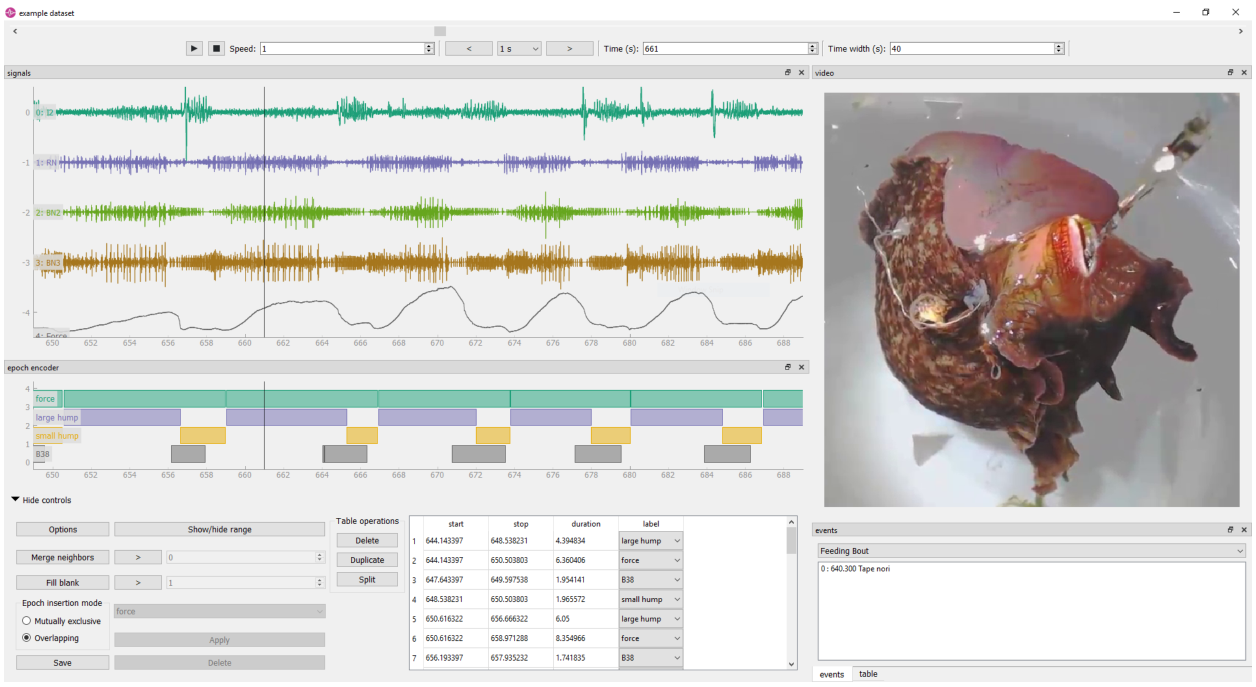
Jul 21, 2019

Table of Contents

Curate, visualize, and annotate your behavioral ephys data using Python

Version: 0.7.0 (other versions)

neurotic is an app that allows you to easily review and annotate your electrophysiology data and simultaneously captured video. It is an easy way to load your Neo-compatible data into ephyviewer without doing any programming. Share a single metadata file with your colleagues and they too will quickly be looking at the same datasets!



Epoch insertion mode: force
( ) Mutually exclusive
(●) Overlapping

	start	stop	duration	label
1	644.143397	648.538231	4.394834	large hump
2	644.143397	650.503803	6.360406	force
3	647.643397	649.597538	1.954141	B38
4	648.538231	650.503803	1.965572	small hump
5	650.616322	656.666322	6.05	large hump
6	650.616322	658.971288	8.354966	force
7	656.193397	657.935232	1.741835	B38

Feeding Bout

0 : 640.300 Tape nori

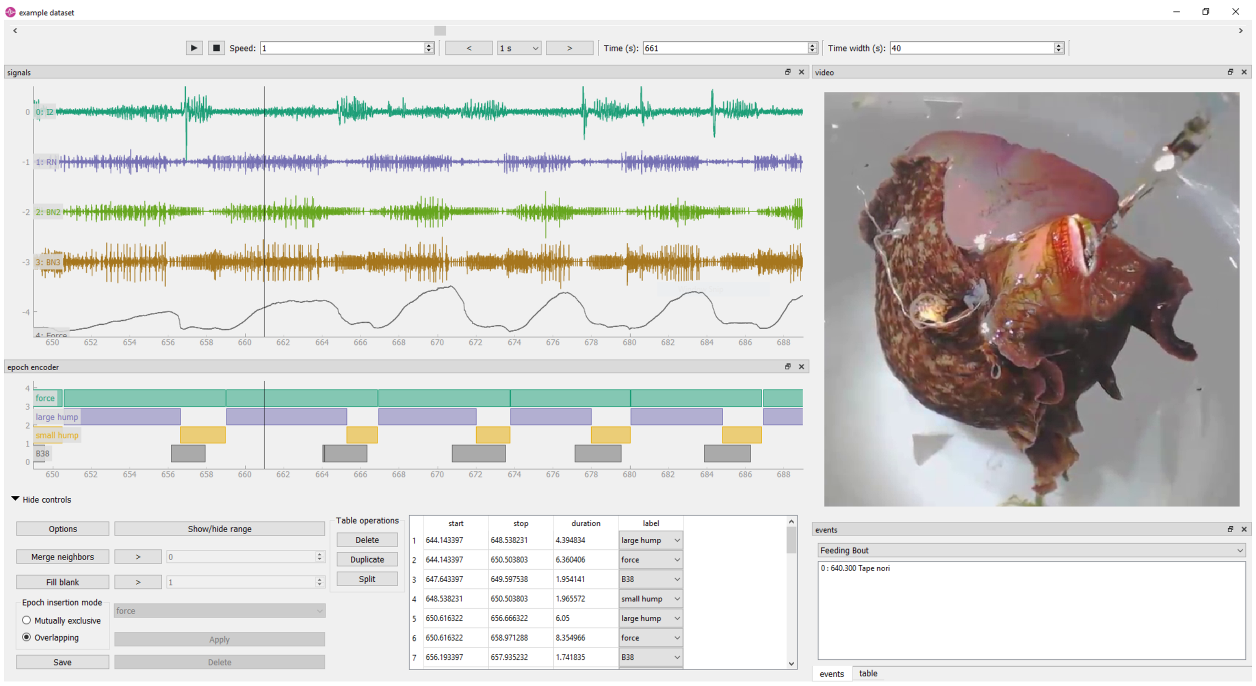
CHAPTER 1

Overview

To use **neurotic**, first organize your datasets in a YAML file like this (see *Configuring Metadata*):

```
my favorite dataset:
    description: This time it actually worked!

    data_dir:           C:/local_dir_containing_files
    remote_data_dir:    http://myserver/remote_dir_containing_downloadable_files  #␣
↪optional
    data_file:          data.axgx
    video_file:         video.mp4
    # etc

    video_offset: -3.4  # seconds between start of video and data acq
    epoch_encoder_possible_labels:
        - label01
        - label02
    plots:
        - channel: I2
          ylim: [-30, 30]
        - channel: RN
          ylim: [-60, 60]
        # etc

    filters:  # used only if fast loading is off (lazy=False)
        - channel: Force
          lowpass: 50
        # etc
    amplitude_discriminators:  # used only if fast loading is off (lazy=False)
        - name: B3 neuron
          channel: BN2
          amplitude: [50, 100]
        # etc

another dataset:
    # etc
```

Open your YAML metadata file in **neurotic** and choose a dataset. If the data and video files aren't already on your local computer, the app can download them for you, even from a password-protected server. Finally, click launch and the app will use a standard viewer layout to display your data to you using ephyviewer.



Speed: 1 | 1 s | Time (s): 661 | Time width (s): 40

Epoch insertion mode: [✓ Overlapping] force

	start	stop	duration	label
1	644.143397	648.538231	4.394834	large hump
2	644.143397	650.503803	6.360406	force
3	647.643397	649.597538	1.954141	B38
4	648.538231	650.503803	1.965572	small hump
5	650.616322	656.666322	6.05	large hump
6	650.616322	658.971288	8.354966	force
7	656.193397	657.935232	1.741835	B38

events: Feeding Bout

0 : 640.300 Tape nori

(Pictured above is a voracious Aplysia californica in the act of making the researcher very happy.)

The viewers are easy and intuitive to navigate (see User Interface):

- Pressing the play button will scroll through your data and video in real time, or at a higher or lower rate if the speed parameter is changed.
- The arrow/WASD keys allow you to step through time in variable increments.
- Jump to a time by clicking on an event in the event list or a table entry in the epoch encoder.
- To show more or less time at once, right-click and drag right or left to contract or expand time.
- Scroll the mouse wheel in the trace viewer or video viewer to zoom.
- The epoch encoder can be used to block out periods of time during which something interesting is happening for later review or further analysis (saved to a CSV file).
- All panels can be hidden, undocked, stacked, or repositioned on the fly.

Electrophysiologists will find this tool useful even if they don't need the video synchronization feature!

Portability is easy with neurotic! Use relative paths in your metadata file along with a remotely accessible data store such as GIN to make your metadata file fully portable. The same metadata file could be copied to a different computer, and downloaded files will automatically be saved to the right place. Data stores can be password protected and **neurotic** will prompt you for a user name and password. This makes it easy to share the **neurotic** experience with your colleagues!

CHAPTER 2

Installation

neurotic requires Python 3.6 or later. It needs PyAV, which is most easily installed from conda-forge. It also does not explicitly list any of its dependencies within the package metadata[1], so they must be installed manually.

To install PyAV and all other dependencies, use these commands (`pip` may raise a non-fatal error that can be ignored; see[2]):

```
conda install -c conda-forge av
pip install "elephant>=0.6.2" "ephyviewer>=1.1.0" "neo>=0.7.2" numpy packaging pandas␣
↪pylttb pyqt5 pyyaml quantities tqdm
```

Finally, install the latest release version of **neurotic** from PyPI, using

```
pip install -U neurotic
```

or install the latest development version from GitHub using

```
pip install -U git+https://github.com/jpgill86/neurotic.git
```

[1] Before **neurotic** can be configured to automatically install dependencies, an upstream library conflict must be fixed. This should be resolved soon; until then, dependencies can be installed manually.

[2] The following warning may appear during dependency installation but can be ignored because the incompatibility between these versions is trivial: `ERROR: elephant 0.6.2 has requirement neo<0.8.0,<=0.7.1, but you'll have neo 0.7.2 which is incompatible`. This is related to the upstream library conflict previously mentioned.

CHAPTER 3

Getting Started

If you installed **neurotic** into a conda environment, first activate it:

```
conda activate <environment name>
```

Launch the standalone app from the command line:

```
neurotic
```

A simple example is provided. Select the "example dataset", download the associated data (~7 MB), and then click "Launch". See User Interface for help with navigation.

Disabling "Fast loading" before launch will enable additional features including amplitude-threshold spike detection and signal filtering.

The command line interface accepts arguments as well:

```
usage: neurotic [-h] [-V] [--no-lazy] [--thick-traces]
                [--theme {light,dark,original}]
                [file] [dataset]

neurotic lets you curate, visualize, and annotate your behavioral ephys data.

positional arguments:
  file                  the path to a metadata YAML file (default: an example
                        file)
  dataset               the name of a dataset in the metadata file to select
                        initially (default: the first entry in the metadata
                        file)

optional arguments:
  -h, --help            show this help message and exit
  -V, --version         show program's version number and exit
  --no-lazy             do not use fast loading (default: use fast loading)
  --thick-traces        enable support for traces with thick lines, which has
                        a performance cost (default: disable thick line
```

(continues on next page)

(continued from previous page)

```
                      support)
  --theme {light,dark,original}
                      a color theme for the GUI (default: light)
```

CHAPTER 4

Configuring Metadata

To load your data with **neurotic**, you must organized them in one or more YAML files, called *metadata files*.

YAML files are very sensitive to punctuation and indentation, so mind those details carefully! Importantly, the tab character cannot be used for indentation; use spaces instead. There are many free websites that can validate YAML for you.

You may include comments in your metadata file, which should begin with `#`.

4.1 Top-Level Organization

Datasets listed within the same metadata file must be given unique names. Optionally, longer descriptions can be provided too. Details pertaining to each dataset, including the description, are nested beneath the dataset name using indentation. You may need to use double quotes around names, descriptions, or other text if they contains special characters (such as `:` or `#`) or are composed only of numbers (such as a date).

```
experiment 2020-01-01:
    description: Both the name and description will be visible when neurotic loads␣
↪the metadata
    # other details about this dataset will go here

my favorite dataset:
    description: This time it actually worked!
    # other details about this dataset will go here
```

4.2 Specifying Data Locations

Within a dataset's YAML block, paths to data and video files should be provided.

All files associated with a dataset should be collected into a single directory. A path to the local copy of this directory must be provided using the `data_dir` key. You may specify `data_dir` as an absolute path (e.g., `C:\Users\me\folder`) or as a path relative to the YAML file (e.g., `folder`).

Paths to individual files within the dataset are provided using keys listed below. These paths should be given relative to `data_dir`. If `data_dir` is flat (no subdirectories), these should be simply the file names. Only `data_file` is required.

Key	Description
`data_file`	A single Neo-compatible data file (required)
`video_file`	A video file that can be synchronized with `data_file`
`annotations_file`	A CSV file for read-only annotations
`epoch_encoder_file`	A CSV file for annotations writable by the epoch encoder
`tridesclous_file`	A CSV file output by tridesclous's DataIO.export_spikes

Note that the `annotations_file` must contain exactly 4 columns with these headers: "Start (s)", "End (s)", "Type", and "Label".

The `epoch_encoder_file` must contain exactly 3 columns with these headers: "Start (s)", "End (s)", and "Type". (The fourth column is missing because ephyviewer's epoch encoder is currently unable to attach notes to individual epochs; this may be improved upon in the future.)

The `tridesclous_file` is described in more detail in *tridesclous Spike Sorting Results*.

4.3 Remote Data Available for Download

Data files must be stored on the local computer for **neurotic** to load them and display their contents. If the files are available for download from a remote server, **neurotic** can be configured to download them for you to the local directory specified by `data_dir` if the files aren't there already.

Specify the URL to the directory containing the data on the remote server using `remote_data_dir`. **neurotic** expects the local `data_dir` and the `remote_data_dir` to have the same structure and will mirror the `remote_data_dir` in the local `data_dir` when you download data (not a complete mirror, just the specified files).

For an example, consider the following:

```
my favorite dataset:
    data_dir:           C:\Users\me\folder
    remote_data_dir:    http://myserver/remote_folder
    data_file:          data.axgx
    video_file:         video.mp4
```

With a YAML file like this, the file paths `data_file` and `video_file` are appended to `remote_data_dir` to obtain the complete URLs for downloading these files, and they will be saved to the local `data_dir`.

If you have many datasets hosted by the same server, you can specify the server URL just once using the special `remote_data_root` key, which should be given outside of any dataset's YAML block with no indentation. This allows you to provide for each dataset a partial URL to a folder in `remote_data_dir` which is relative to `remote_data_root`. For example:

```
remote_data_root: http://myserver

my favorite dataset:
    data_dir:           C:\Users\me\folder1
    remote_data_dir:    remote_folder1
    data_file:          data.axgx
    video_file:         video.mp4

```

(continues on next page)

(continued from previous page)

```
another dataset:
    data_dir:           C:\Users\me\folder2
    remote_data_dir:    remote_folder2
    data_file:          data.axgx
    video_file:         video.mp4
```

Here, URLs to video files are composed by joining `remote_data_root` + `remote_data_dir` + `video_file`.

Recall that if `data_dir` is a relative path, it is assumed to be relative to the YAML file. In the example above, if the YAML file is located in `C:\Users\me`, the paths could be abbreviated:

```
remote_data_root: http://myserver

my favorite dataset:
    data_dir:           folder1
    remote_data_dir:    remote_folder1
    data_file:          data.axgx
    video_file:         video.mp4

another dataset:
    data_dir:           folder2
    remote_data_dir:    remote_folder2
    data_file:          data.axgx
    video_file:         video.mp4
```

Note: Portability is easy with neurotic! Use relative paths in your metadata file along with a remotely accessible data store such as GIN to make your metadata file fully portable. The same metadata file could be copied to a different computer, and downloaded files will automatically be saved to the right place. Data stores can be password protected and **neurotic** will prompt you for a user name and password. This makes it easy to share the **neurotic** experience with your colleagues!

4.4 Video Synchronization Parameters

4.4.1 Constant Offset

If data acquisition began with some delay after video capture began, provide a negative value for `video_offset` equal to the delay in seconds. If video capture began after the start of data acquisition, use a positive value. A value of zero will have no effect.

neurotic warns users about the risk of async if `video_file` is given but `video_offset` is not. To eliminate this warning for videos that have no delay, provide zero.

4.4.2 Frame Rate Correction

If the average frame rate reported by the video file is a little fast or slow, you may notice your video and data going out of sync late in a long experiment. You can provide the `video_rate_correction` parameter to fix this. The reported average frame rate of the video file will be multiplied by this factor to obtain a new frame rate used for playback. A value less than 1 will decrease the frame rate and shift video events to later times. A value greater than 1 will increase the frame rate and shift video events to earlier times. A value of 1 has no effect.

You can obtain a good estimate of what value to use by taking the amount of time between two events in the video and dividing by the amount of time between the same two events according to the data record (seen, for example, as synchronization pulses or as movement artifacts).

4.4.3 Discrete Desynchronization Events

If you paused data acquisition during your experiment while video capture was continuous, you can use the `video_jumps` parameter to correct for these discrete desynchronization events, assuming you have some means of reconstructing the timing. For each pause, provide an ordered pair of numbers in seconds: The first is the time *according to data acquisition* (not according to the video) when the pause occurred, and the second is the duration of the pause during which the video kept rolling.

For example:

```
my favorite dataset:
    video_file: video.mp4
    # etc

    video_jumps:
        # a list of ordered pairs containing:
        # (1) time in seconds when paused occurred according to DAQ
        # (2) duration of pause in seconds
        - [60, 10]
        - [120, 10]
        - [240, 10]
```

These values could correct for three 10-second pauses occurring at times 1:00, 2:00, 3:00 according to the DAQ, which would correspond to times 1:00, 2:10, 3:20 according to the video. The extra video frames captured during the pauses will be excised from playback so that the data and video remain synced.

neurotic will automatically suggest values for `video_jumps` if it reads an AxoGraph file that contains stops and restarts (only if `video_jumps` is not already specified).

4.5 Plot Parameters

Use the `plots` parameter to specify which signal channels from `data_file` you want plotted and how to scale them.

Consider the following example, and notice the use of hyphens and indentation for each channel.

```
my favorite dataset:
    data_file: data.axgx
    # etc

    plots:
        - channel: Extracellular
          ylabel: Buccal nerve 2 (BN2)
          units: uV
          ylim: [-150, 150]

        - channel: Intracellular
          ylabel: B3 neuron
          units: mV
          ylim: [-100, 50]
```

(continues on next page)

(continued from previous page)

```
        - channel: Force
          units: mN
          ylim: [-10, 500]
```

This would plot the "Extracellular", "Intracellular", and "Force" channels from the `data_file` in the given order. `ylabel` is used to relabel a channel and is optional. The `units` and `ylim` parameters are used together to scale each signal such that the given range fits neatly between the traces above and below it. If `units` is not given, they are assumed to be microvolts for voltage signals and millinewtons for force signals. If `ylim` is not given, they default to `[-120, 120]` for voltages and `[-10, 300]` for forces.

If `plots` is not provided, all channels are plotted using the default ranges, except for channels that match these patterns: "Analog Input #*" and "Clock". Channels with these names can be plotted if given explicitly by `plots`.

The amount of time initially visible can be specified in seconds with `t_width`.

4.6 Epoch Encoder Parameters

The labels available to the epoch encoder must be specified ahead of time using `epoch_encoder_possible_labels` (this is a current limitation of ephyviewer that may eventually be improved upon).

For example:

```
my favorite dataset:
    epoch_encoder_file: epoch-encoder.csv
    # etc

    epoch_encoder_possible_labels:
        - label1
        - label2
        - label3
```

4.7 Filters

Highpass, lowpass, and bandpass filtering can be applied to signals using the `filters` parameter. Note that filters are only applied if fast loading is off (`lazy=False`).

Consider the following example, and notice the use of hyphens and indentation for each filter.

```
my favorite dataset:
    data_file: data.axgx
    # etc

    filters:  # used only if fast loading is off (lazy=False)

        - channel: Extracellular
          highpass: 300 # Hz
          lowpass: 500 # Hz

        - channel: Intracellular
          highpass: 300 # Hz
```

(continues on next page)

(continued from previous page)

```
        - channel: Force
          lowpass: 50 # Hz
```

Filter cutoffs are given in hertz. Combining `highpass` and `lowpass` provides bandpass filtering.

4.8 Amplitude Discriminators

Spikes with peaks that fall within amplitude windows given by `amplitude_discriminators` can be automatically detected by **neurotic** on the basis of amplitude alone. Note that amplitude discriminators are only applied if fast loading is off (`lazy=False`).

Detected spikes are indicated on the signals with markers, and spike trains are displayed in a raster plot.

In addition to restricting spike detection for a given unit to an amplitude window, detection can also be limited in time to overlap with epochs with a given label.

Consider the following example, and notice the use of hyphens and indentation for each amplitude discriminator.

```
my favorite dataset:
    data_file: data.axgx
    # etc

    amplitude_discriminators:  # used only if fast loading is off (lazy=False)

        - name: Unit 1
          channel: Extracellular
          amplitude: [50, 150] # uV

        - name: Unit 2
          channel: Extracellular
          amplitude: [20, 50] # uV
          epoch: Unit 2 activity
```

Here two units are detected on the same channel with different amplitude windows. Any peaks between 50 and 150 microvolts on the “Extracellular” channel will be tagged as a spike belonging to “Unit 1”. The discriminator for “Unit 2” provides the optional `epoch` parameter. This restricts detection of “Unit 2” to spikes within the amplitude window that occur at the same time as epochs labeled “Unit 2 activity”. These epochs can be created by the epoch encoder (reload required to rerun spike detection at launch-time), specified in the read-only `annotations_file`, or even be contained in the `data_file` if the format supports epochs.

Amplitude windows are permitted to be negative.

4.9 tridesclous Spike Sorting Results

tridesclous is a sophisticated spike sorting toolkit. The results of a sorting process can be exported to a CSV file using tridesclous’s DataIO.export_spikes function. This file contains two columns: the first is the sample index of a spike, and the second is the ID for a cluster of spikes. If this file is specified with `tridesclous_file`, then a mapping from the cluster IDs to channels must be provided with `tridesclous_channels`.

In the following example, notice the lack of hyphens:

```
my favorite dataset:
    data_file: data.axgx
    tridesclous_file: spikes.csv
    # etc

    tridesclous_channels:
        0: [Channel A, Channel B]
        1: [Channel A]
        2: [Channel B]
        3: [Channel B]
        # etc
```

Here numeric cluster IDs are paired with a list of channels found in `data_file` on which the spikes were detected.

To show only a subset of clusters or to merge clusters, add the `tridesclous_merge` parameter.

In this example, note again the punctuation:

```
my favorite dataset:
    data_file: data.axgx
    tridesclous_file: spikes.csv
    # etc

    tridesclous_channels:
        0: [Channel A, Channel B]
        1: [Channel A]
        2: [Channel B]
        3: [Channel B]
        # etc

    tridesclous_merge:
        - [0, 1]
        - [3]
```

Now clusters 0 and 1 are combined into a single unit, and only that unit and cluster 3 are plotted; cluster 2 has been discarded.

4.10 A Complete Example

These are the contents of the example metadata file that ships with **neurotic**, which can be loaded by running `neurotic` from the command line without arguments:

```
example dataset:
    description: This is an example data set

    # these data are a subset of Jeffrey Gill's dataset 2018-06-21_IN-VIVO_JG-08 002
    data_dir:            example-data
    remote_data_dir:     https://web.gin.g-node.org/jpgill86/neurotic-data/raw/master/
↪examples/example-data
    data_file:           data.axgx
    video_file:          video.mp4
    annotations_file:    annotations.csv
    epoch_encoder_file: epoch-encoder.csv

    video_offset: 640.3 # seconds
```

(continues on next page)

(continued from previous page)

```
    epoch_encoder_possible_labels:
        - force
        - large hump
        - small hump
        - B38

    plots:
        - channel: I2
          units: uV
          ylim: [-30, 30]

        - channel: RN
          units: uV
          ylim: [-60, 60]

        - channel: BN2
          units: uV
          ylim: [-120, 120]

        - channel: BN3
          units: uV
          ylim: [-150, 150]

        - channel: Force
          units: mN
          ylim: [-10, 300]

    filters:  # used only if fast loading is off (lazy=False)

        - channel: I2
          lowpass: 100 # Hz

        - channel: Force
          lowpass: 50 # Hz

    amplitude_discriminators:  # used only if fast loading is off (lazy=False)

        - name: B3
          channel: BN2
          amplitude: [50, 150] # uV

        - name: B38
          channel: BN2
          amplitude: [17, 26] # uV
          epoch: B38

        - name: B4/B5
          channel: BN3
          amplitude: [85, 200] # uV
```

CHAPTER 5

Release Notes

5.1 neurotic 0.7.0

2019-07-21

5.1.1 Improvements

- New documentation hosted at Read the Docs: https://neurotic.readthedocs.io
- Add menu action for opening metadata in editor (#83)
- Add menu action for opening the selected data directory (#84)
- Add list of installed versions of dependencies and doc links to About window (#44, #65)

5.1.2 Bug fixes

- Fix files remaining locked after closing a fast-loaded window (#69)
- Fix launching from command line with bad metadata argument (#82)

5.2 neurotic 0.6.0

2019-07-10

5.2.1 Improvements

- Add a basic "About neurotic" window with version and website information (#38)
- Update logo (#39)

- Add keywords and project URLs to package metadata (#40)

5.3 neurotic 0.5.1

2019-07-09

5.3.1 Compatibility updates

- Compatibility update for RawIOs with non-zero offset (#37)

5.4 neurotic 0.5.0

2019-07-06

5.4.1 Improvements

- Support fast (lazy) loading in Neo < 0.8.0 (#35)
- Add "git." and conditionally ".dirty" to dev local version identifier (#34)

5.5 neurotic 0.4.2

2019-07-06

5.5.1 Bug fixes

- Fix for EstimateVideoJumpTimes regression introduced in 0.4.0 (#33)

5.6 neurotic 0.4.1

2019-07-02

5.6.1 Compatibility updates

- Change sources of development versions of dependencies (#32)
- Compatibility update for scaling of raw signals (#31)

5.7 neurotic 0.4.0

2019-07-01

5.7.1 Improvements

- Show epochs imported from CSV files with zero duration in epoch viewer (#27)
- Show epochs/events imported from data file in events list/epoch viewer (#28)
- Alphabetize epoch and event channels by name (#29)

5.8 neurotic 0.3.0

2019-06-29

5.8.1 Improvements

- Remove dependency on ipywidgets by making notebook widgets optional (#25)
  - Notebook widget classes renamed: `MetadataSelector` → `MetadataSelectorWidget`, `EphyviewerConfigurator` → `EphyviewerConfiguratorWidget`
- Add app description and screenshot to README (#22)
- Promote to beta status (#23)

5.9 neurotic 0.2.0

2019-06-28

5.9.1 Improvements

- Add basic command line arguments (#14)
- Add continuous integration with Travis CI for automated testing (#13)
- Add some tests (#15, #16)
- Migrate example data to GIN (#18)

5.9.2 Bug fixes

- Fix crash when downloading from a server that does not report file size (#17)
- Raise an exception if a Neo RawIO cannot be found for the data file (#12)

5.10 neurotic 0.1.1

2019-06-22

5.10.1 Bug fixes

- Fix various downloader errors (#7)

5.11 neurotic 0.1.0

2019-06-22

- First release