SpikeDB User Manual

Brandon Aubie

Last Updated: March 1 2011

Contents

Chapter 1

Introduction

SpikeDB is a database and analysis tool for electrophysiological recordings done with Spike. It runs on Linux and Mac (and probably Windows).

Chapter 2

Basic User Interface

2.1 Browse Files

2.1.1 Plots

General Usage

- **Zoom** - Left click and drag horizontally to zoom in on a subsection of data.
- **Export Data** - Right click anywhere on the plot to bring up the options menu and select `Export Data`. This allows you to export the plotted data in CSV files that are ready for import into Excel or for use in other graphing software such as GLE.

Spike Raster

The spike raster is a built in plot that displays the stimuli as red (channel 1) and blue (channel 2) lines and spikes as black dots. Zooming in on this plot will limit the spike times available to the quick analysis plot on the right.

Quick Analysis

By default, this plot will display the mean number of spikes per trial. Other analysis plugins are available in the drop down box or additional plugins can be loaded by clicking the `Open` icon. Generally, it is wise to use plugins that operate on selected files only here as no text display is available. For more general analysis on many files use the Analysis tab. That said, if multiple files of the same type are selected, the plots can be overlaid for easy comparison.

Chapter 3

Analysis Plug-In Module

3.1 Basic Usage

The Analysis Plug-In Module allows you to use the Python scripting language write custom analysis routines on one or many Spike recording files. Each Python script will have the `SpikeDB` object available to it. This object provides access to all of the data held within SpikeDB as well as a host of methods useful for analysis.

3.2 Reference

All functions listed below are accessed via the SpikeDB object. For example, `SpikeDB.getFiles(True)` calls the `getFiles` method.

3.2.1 `void filterSpikesAbs(float minSpikeTime,float maxSpikeTime)`

Parameters

`minSpikeTime`	Minimum absolute spike time.
`maxSpikeTime`	Maximum absolute spike time.

Description

Filters the spikes for every file based on the absolute time of the spike in the file.

3.2.2 `void filterSpikesRel(float minSpikeTime,float maxSpikeTime)`

Parameters

`minSpikeTime`	Minimum relative spike time.
`maxSpikeTime`	Maximum relative spike time.

Description

Filters the spikes for every file based on the time of the spike relative to the stimuli onset and offsets. A spike is included only if it falls within a stimulus onset+`minSpikeTime` and stimulus offset+`maxSpikeTime`.

When both channel 1 and channel 2 are active a spike is included if it falls within the relative timing of either stimulus. To include spikes prior to stimulus onset set `minSpikeTime` to a value less than zero.

3.2.3 `list getCells()`

3.2.4 `list getFiles(bool onlySelected)`

3.2.5 `float mean(list values)`

3.2.6 `void plotClear()`

3.2.7 `void plotLine(list xValues,list yValues,list errValues)`

3.2.8 `void plotSetLineWidth(float lineWidth)`

3.2.9 `void plotSetPointSize(float pointSize)`

3.2.10 `void plotSetRGBA(float red,float green,float blue,float alpha)`

3.2.11 `void plotXLabel(string XLabel)`

3.2.12 `void plotXMin(float minXValue)`

3.2.13 `void plotXMax(float maxXValue)`

3.2.14 `void plotYLabel(string YLabel)`

3.2.15 `void plotYMin(float minYValue)`

3.2.16 `void plotYMax(float maxYValue)`

3.2.17 `void plotYMax(float maxYValue)`

3.2.18 `float stddev(list values)`

3.2.19 `void write(string text)`

Parameters

`text`	Text to print to SpikeDB output window.

Description

This function is used internally to print text to the SpikeDB output window and is generally not needed by analysis script writers. Standard Python output functions like `print` work just fine and print to the SpikeDB output window as expected. Errors are also printed to the SpikeDB window. Note that the output window is only available in the Analysis tab and not in the quick analysis plot.

3.3 Examples

3.3.1 Mean Spike Times

Listing 3.1: Calculating the mean spike counts.

```
files = SpikeDB.getFiles(True)
```

```
for f in files:
        means = []
        err = []
        x = []
        for t in f[’trials’]:
                count = []
                x.append(t[’xvalue’])
                for p in t[’spikes’]:
                        count.append(len(p))
                means.append(SpikeDB.mean(count))
                err.append(SpikeDB.stddev(count))
        SpikeDB.plotXLabel(f[’xvar’])
        SpikeDB.plotYLabel(’Mean␣Spike␣Count’)
        SpikeDB.plotLine(x,means,err)
```