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.

Quick Analysis Plugins

Several Quick Analysis plugins are included with SpikeDB, however, it is easy to add your own. Quick Analysis are shown in the drop-down box on the Quick Analysis and Analysis toolbars and are defined by scripts located in the plugins/ folder of the SpikeDB application. The location of this folder depends on your operating system. Refer to Table 3.1 for details. Quick Analysis plugins function exactly the same way as

Table 3.1: Default plugins folder locations on different operating systems.

Microsoft Windows	C:\Program Files\SpikeDB\plugins\
Mac OS X	/Applications/SpikeDB.app/Contents/Resources/plugins/ (built-in)
or	/Library/Application Support/plugins/ (user added)
Linux	/usr/share/SpikeDB/plugins/

other Analysis plugins with two minor differences. First, the name of the plugin to display in the toolbar drop-down list must be specified on the first line with three # symbols in the format

```
### Name Goes Here
```

Second, when retrieving files with the `getFiles()` method you will generally always want to pass `True` as the parameter to ensure that only the currently selected file(s) will be analyzed. When adding new Quick Analysis plugins to the plugins folder you must restart SpikeDB for them to be available in the drop-down box.

3.2 Reference

All functions listed in this section 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.

Listing 3.1: Example

```
# Have getFiles() only show spikes
# that occured between 10 and 50 ms.
SpikeDB.filterSpikesAbs(10, 50)
```

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.

Listing 3.2: Example

```
# Have getFiles() only show spikes
# that occured 5 ms before and 50 ms after
# a given stimulus.
SpikeDB.filterSpikesRel(-5, 50)
```

3.2.3 `list` getCells()

Description

A list of dictionary objects is returned where each dictionary object represents a single cell. All cells presently displayed in the files list on the browse page are always returned. The structure of each dictionary object is shown in Table 3.2.

Table 3.2: Dictionary structure for each cell in the list of cells returned by getCells().

'AnimalID'	The animal ID where the cell was found.
'CellID'	The ID of the cell.
'CarFreq'	The cell's carrier frequency as manually entered in the cell details window.
'Threshold'	The cell's threshold as manually entered in the cell details window.
'Depth'	The cell's depth as manually entered in the cell details window.

Listing 3.3: Example

```
# Return a list of all cells in the browse files list.
cells = SpikeDB.getCells()
```

3.2.4 `list` getFiles(`bool onlySelected`)

Parameters

onlySelected	When TRUE, only return a list of the files selected in the files list on the browse page. This is useful when writing a Quick Analysis plugin.

Description

A list of dictionary objects is returned where each dictionary object represents a single file. The structure of each dictionary object is shown in Table 3.3. Note that a "trial" is a value of a stimulus at a particular X-variable value. For example, if a file varied the stimulus duration then a trial contains all of the passes for a particular stimulus duration.

Listing 3.4: Example

```
# Return a list of all files in the browse files list.
allFiles = SpikeDB.getFiles(False)

# Return a list of selected files in the browse files list.
selFiles = SpikeDB.getFiles(True)
```

Table 3.3: Dictionary structure for each cell in the list of cells returned by getFiles().

'AnimalID'		The animal ID where the cell was found.
'CellID'		The ID of the cell.
'FileID'		The ID of the file.
'datetime'		A string in the form YYYY-MM-DD HH:MM:SS of the time the file was created.
'xvar'		String representation of the X variable.
'type'	1	Stimulus type (Sinus, Swept Sinus, FM, etc.) on channel 1. Blank if none.
	2	Stimulus type (Sinus, Swept Sinus, FM, etc.) on channel 2. Blank if none.
'duration'	1	Stimulus duration on channel 1. SpikeDB.VARYING if varied.
	2	Stimulus duration on channel 2. SpikeDB.VARYING if varied.
'attenuation'	1	Stimulus attenuation on channel 1. SpikeDB.VARYING if varied.
	2	Stimulus attenuation on channel 2. SpikeDB.VARYING if varied.
'frequency'	1	Stimulus frequency on channel 1. SpikeDB.VARYING if varied.
	2	Stimulus frequency on channel 2. SpikeDB.VARYING if varied.
'begin'	1	Stimulus start time on channel 1. SpikeDB.VARYING if varied.
	2	Stimulus start time on channel 2. SpikeDB.VARYING if varied.
'trials'		List containing dictionary objects defined by:

	'xvalue'		Value of the X variable for this trial.
	'duration'	1	Stimulus duration on channel 1.
		2	Stimulus duration on channel 2.
	'attenuation'	1	Stimulus attenuation on channel 1.
		2	Stimulus attenuation on channel 2.
	'frequency'	1	Stimulus frequency on channel 1.
		2	Stimulus frequency on channel 2.
	'begin'	1	Stimulus start time on channel 1.
		2	Stimulus start time on channel 2.
	'passes'		A list of lists of spike times. Each list represents a different pass over this stimulus parameter (i.e. in this trial). For example, the 'passes' list for a trial with 4 passes could look like [[23.13,24,9], [22.1], [], [24.22]].

3.2.5 `float` mean(`list values`)

Parameters

`values`	List of numbers to calculate the mean of.

Description

Returns the mean value of all the numbers passed in the `values` list.

Listing 3.5: Example

```
vals = [1,2,3,4]
mean  = SpikeDB.mean(vals)
print mean
# Output:
#    2.5
```

3.2.6 `void` `plotClear()`

Description

Clears the SpikeDB plot and resets the plot variables and style settings. This is called automatically at the top of each script so generally should not be needed.

Listing 3.6: Example

```
# Clear the plot
SpikeDB.plotClear()
```

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

Parameters

xValues	List of the X values for the (X,Y) points to plot.
yValues	List of the Y values for the (X,Y) points to plot.
errValues	List of the magnitude of the error bars. Enter an empty list for no error bars.

Description

Plot a series of (X,Y) points in the style determined by prior plot setup functions (i.e. plotSetLineWidth, plotSetPointSize, plotSetRGBA, etc.). The length of xValues and yValues must be the same and errValues must also be the same length or be empty (i.e. []). In some cases it may not make sense to plot a point for a particular X value (for example, there is no defined first spike latency for a trial with zero spikes). In such cases you can assign SpikeDB.NOPOINT to that Y value to produce a disconnected line missing that point.

Listing 3.7: Example

```
X = [1,2,3,4,5]
Y = [1,4,SpikeDB.NOPOINT,16,25]
err = [0.5,0.2,0,1.1,0.3]

# Plot the data with error bars
SpikeDB.plotLine(X,Y,err)

# Plot the data without error bars
SpikeDB.plotLine(X,Y,[])
```

3.2.8 `void` `plotSetLineWidth(`float lineWidth`)`

Parameters

`lineWidth`	The line width connecting points in a plot. Default is 2.

Description

Determines the width of the line when plotting data points for the next call to `plotLine()`. To remove lines, use a line width of 0.

Listing 3.8: Example

```
X = [1,2,3,4,5]
Y = [1,4,9,16,25]
err = [0.5,0.2,0.9,1.1,0.3]

# Remove the line connecting points
SpikeDB.plotSetLineWidth(0)

# Plot the data
SpikeDB.plotLine(X,Y,err)
```

3.2.9 `void plotSetPointSize(float pointSize)`

Parameters

`pointSize`	The point size for points in a plot. Default is 8.

Description

Determines the size of point shapes when plotting data points for the next call to `plotLine()`.

Listing 3.9: Example

```
X = [1,2,3,4,5]
Y = [1,4,9,16,25]
err = [0.5,0.2,0.9,1.1,0.3]

# Use very large points.
SpikeDB.plotSetPointSize(16)

# Plot the data
SpikeDB.plotLine(X,Y,err)
```

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

Parameters

red	Percentage of red between 0 and 1.
green	Percentage of green between 0 and 1.
blue	Percentage of blue between 0 and 1.
alpha	Percentage of alpha between 0 (opaque) and 1 (transparent).

Description

Determine the color of the points and line for the next call to plotLine().

Listing 3.10: Example

```
X = [1,2,3,4,5]
Y = [1,4,9,16,25]
err = [0.5,0.2,0.9,1.1,0.3]

# Draw in semi-translucent red.
SpikeDB.plotSetRGBA(1,0,0,0.25)

# Plot the data
SpikeDB.plotLine(X,Y,err)
```

3.2.11 void plotXLabel(string xLabel)

Parameters

`xLabel`	Text to show under the x-axis.

Description

Determine the x-axis label for the next call to `plotLine()`.

```
X = [1,2,3,4,5]
Y = [1,4,9,16,25]
err = [0.5,0.2,0.9,1.1,0.3]

# Set the labels.
SpikeDB.plotXLabel('X Value')
SpikeDB.plotYLabel('Squared Value')

# Plot the data
SpikeDB.plotLine(X,Y,err)
```

16

3.2.12 void plotXMin(float minXValue)

Parameters

`minXValue`	Minimum value to show on the x-axis.

Description

Use this function to force the plot to have the x-axis begin at `minXValue`.

Listing 3.12: Example

```
X = [1,2,3,4,5]
Y = [1,4,9,16,25]
err = [0.5,0.2,0.9,1.1,0.3]

# Constrain the plot
SpikeDB.plotXMin(2)
SpikeDB.plotXMax(4)
SpikeDB.plotYMin(3)
SpikeDB.plotYMax(17)

# Plot the data
SpikeDB.plotLine(X,Y,err)
```

3.2.13   void plotXMax(float maxXValue)

Parameters

`maxXValue`	Maximum value to show on the x-axis.

Description

Use this function to force the plot to have the x-axis end at `maxXValue`.

```
X = [1,2,3,4,5]
Y = [1,4,9,16,25]
err = [0.5,0.2,0.9,1.1,0.3]

# Constrain the plot
SpikeDB.plotXMin(2)
SpikeDB.plotXMax(4)
SpikeDB.plotYMin(3)
SpikeDB.plotYMax(17)

# Plot the data
SpikeDB.plotLine(X,Y,err)
```

3.2.14 void plotYLabel(string yLabel)

Parameters

`yLabel`	Text to show beside the y-axis.

Description

Determine the y-axis label for the next call to `plotLine()`.

Listing 3.14: Example

```
X = [1,2,3,4,5]
Y = [1,4,9,16,25]
err = [0.5,0.2,0.9,1.1,0.3]

# Set the labels.
SpikeDB.plotXLabel('X Value')
SpikeDB.plotYLabel('Squared Value')

# Plot the data
SpikeDB.plotLine(X,Y,err)
```

3.2.15 void plotYMin(float minYValue)

Parameters

minYValue	Minimum value to show on the y-axis.

Description

Use this function to force the plot to have the y-axis begin at minYValue.

Listing 3.15: Example

```
X = [1,2,3,4,5]
Y = [1,4,9,16,25]
err = [0.5,0.2,0.9,1.1,0.3]

# Constrain the plot
SpikeDB.plotXMin(2)
SpikeDB.plotXMax(4)
SpikeDB.plotYMin(3)
SpikeDB.plotYMax(17)

# Plot the data
SpikeDB.plotLine(X,Y,err)
```

3.2.16 void plotYMax(float maxYValue)

Parameters

maxYValue	Maximum value to show on the y-axis.

Description

Use this function to force the plot to have the y-axis end at maxYValue.

Listing 3.16: Example

```
X = [1,2,3,4,5]
Y = [1,4,9,16,25]
err = [0.5,0.2,0.9,1.1,0.3]

# Constrain the plot
SpikeDB.plotXMin(2)
SpikeDB.plotXMax(4)
SpikeDB.plotYMin(3)
SpikeDB.plotYMax(17)

# Plot the data
SpikeDB.plotLine(X,Y,err)
```

3.2.17 `float stddev(list values)`

Parameters

`values`	List of numbers to calculate the standard deviation of.

Description

Returns the standard deviation of all the numbers passed in the `values` list. The function assumes you are calculating the sample standard deviation and thus the following formula is used:

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})^2}$$

Listing 3.17: Example

```
vals = [1,2,3,4]
sd   = SpikeDB.stddev(vals)
print sd
# Output:
#    1.66666666667
```

3.2.18 `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 Complete Examples

3.3.1 Mean Spike Times

Listing 3.18: Calculating the mean spike counts.

```
### Mean Spike Count

# Get selected files only
files = SpikeDB.getFiles(True)

# Plot means for each file.
for f in files:

        # Create placeholder lists
        means = []
        err = []
        x = []

        # Calculate the mean and standard deviation
        # for each trial in the file
        for t in f['trials']:

                # Placeholder for the spike counts
                count = []

                # Get the X value
                x.append(t['xvalue'])

                # Get the spike counts for each pass
                for p in t['passes']:
                        count.append(len(p))

                # Calculate the mean and standard devations
                means.append(SpikeDB.mean(count))
                err.append(SpikeDB.stddev(count))

        # Plot this file
        SpikeDB.plotXLabel(f['xvar'])
        SpikeDB.plotYLabel('Mean Spike Count')
        SpikeDB.plotLine(x,means,err)
```