MLPy – An implementation of the Maximum Likelihood Procedure in Python

Floris van Vugt
f.t.vanvugt[AT]gmail(dot)com
University Lyon-1, France
University of Music and Drama, Hanover, Germany

MLPy is intended to be a quick-and-dirty implementation of the Maximum Likelihood Procedure to establish psychophysical thresholds. It was first proposed by Green (1993 JASA) and Gu & Green (1994 JASA).

MLPy is heavily inspired by the seminal Matlab MLP toolbox by Massimo Grassi, and it does not even come close to this latter in terms of arsenal of available tests and completeness. My intention with MLPy was to write a module that was simple in use, flexible and that could be used on computers without Windows or Matlab – since at this time I'm doing clinical work where we have computers running Linux and no Matlab.

What is MLP?
For an in-depth explanation, the paper Green (1993 JASA) is an excellent and accessible reference. Also, an excellent overview is Grassi & Soranzo (2009 Behav Res Methods). In short, the idea of MLP is to entertain a large number of hypothetical psychometric curves. Then, every time the participant responds to a stimulus, we calculate how likely his or her entire set of responses is, given each of the psychometric curves. Then we choose the psychometric curve that makes the actual observations most likely (i.e. *maximum likelihood estimate*). We then use this estimate psychometric curve to calculate where is the best next stimulus level to put to the participant (the *sweetpoint*). In this way, MLP is shown to be faster than classical staircase procedures. But you have to be careful, some assumptions go in to the algorithm.

How does MLPy work?
You can use MLP to measure a variety of different types of thresholds. For example, what is the smallest difference in volume that you can detect? What is the lowest sound volume you can hear? Currently, however, MLPy only implements one threshold detection: that of distinguishing a isochronous from a non-isochronous sequence of tones. This is used by Hyde & Peretz (2004 Psych Sci). We recreate these stimuli, with sine tones instead of piano tones.
To run one block of trials, simply run:

```
python mlp.py
```

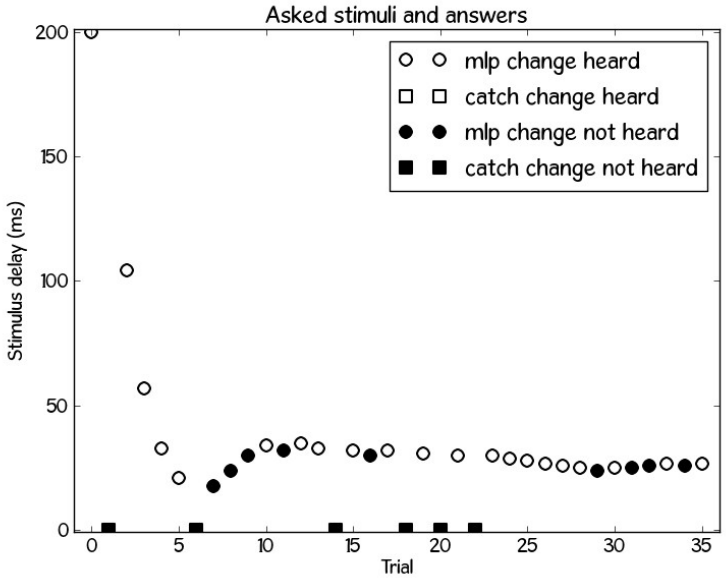
After 30 trials (if you use the default settings), it will return to you an estimate of the center of your psychometric curve in hearing the difference between an isochronous and non-isochronous five-tone rhythm. At the same time, it will output two data files.
You can then run:

```
python analyse_mlp_result.py
```

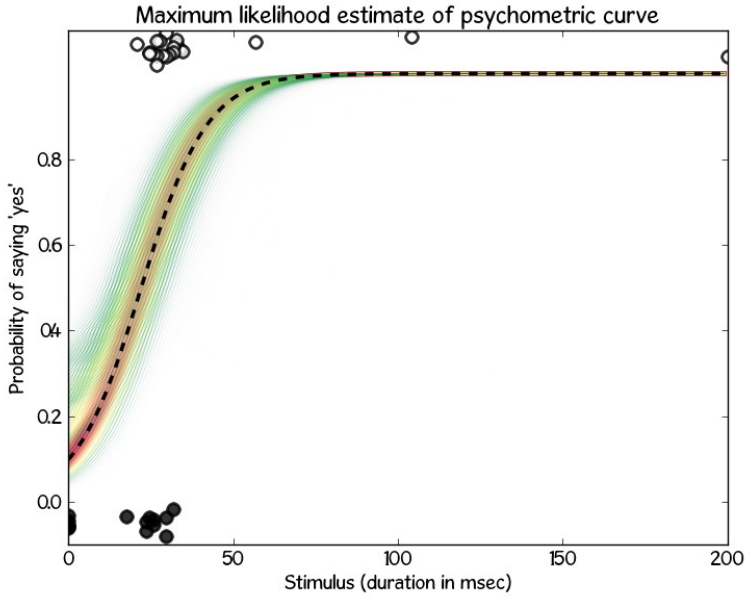
This will prompt you to select a data file and then show you a number of visualisations. Here are some samples to whet your appetite.

Data visualisations

In the first graph, you'll see the stimulus intensities asked at each instance. White circles indicate a "yes" answer, and black circles a "no" answer:



So what is the estimated threshold? In the graph below you can see the different hypothetical psychometric curves. The colour coding indicates their likelihood estimate: curves that make the data very likely are more red, whereas those that make it less likely are green. The ones that make the data very unlikely are no longer visible. The little dots on the top (above 1.0) indicate the stimuli for which the answer was "yes", with some random vertical jitter so you get a better idea of the density of points. The little dots on the bottom do the same for the "no" answers.



Data files
Two data files are generated when you run the script. They carry the name of the participant and a timestamp. One is the meta-file and the other is the raw data file.

The raw data file has the following format:

```
PARTICIPANT TRIAL TYPE STIMULUS RESPONSE
floris 0 mlp 200.000000 1
floris 1 catch 0.000000 0
floris 2 mlp 104.070555 1
floris 3 mlp 56.834374 1
floris 4 mlp 32.713771 1
floris 5 mlp 20.653470 1
... etc.
```

You can open this in a spreadsheet editor or in a text editor. As you can see, it will list each trial separately, with the stimulus level (in our case a number of milliseconds delay). The response is what the participant responded (0=no change heard, 1=change heard).

But what is the threshold? To be able to determine this, you can open the meta file. It looks as follows:

```
participant          : floris
slope                : 0.100000
min-hyp              : 0.000000
max-hyp              : 200.000000
n.hypotheses         : 200
false-alarm-rates    : 0.00,0.10,0.20,0.30
target p             : 0.612375
n.trials.A           : 10
n.catch trials.A     : 2
n.trials.B           : 20
n.catch trials.B     : 4
initial stim         : 200.000000


Maximum likelihood estimate: 22.11
False alarm rate estimate: 0.00
```

As you can see, it gives a plain overview of the parameters we've used, so that you don't forget them later on. The last two lines indicate the estimated threshold value, as well as the estimated false alarm rate.

Technical details
The `mlp.py` script calls `toneseries.py` which is in charge of generating the Hyde & Peretz (2004) stimuli. These are saved as a temporary wave file in the working directory (`stim.wav`). The file is then played using a command-line tool. In MacOS, afplay does this. In Linux, aplay does this. Under Windows you can use any tool.
You could also bypass the stage where the wave file gets written, and simply play the binary signal as it's generated in `toneseries.py`. I've not used this option, since it will depend on the platform and

sound system how you want to do this (have a look here: http://stackoverflow.com/questions/307305/play-a-sound-with-python).

What do you need?
Python. You can download it for free on www.python.org. You will also need numpy, which is available here: numpy.scipy.org. If you want to use the visualisations, you will also need matplotlib: matplotlib.sourceforge.net.

I hope this little module will be of help to you. Please feel free to modify it, add things to it. If you add something that you think could be of use for others, don't hesitate to e-mail me, and I'll be happy to include it in a future release.