# Filtering Data in the Audio Domain

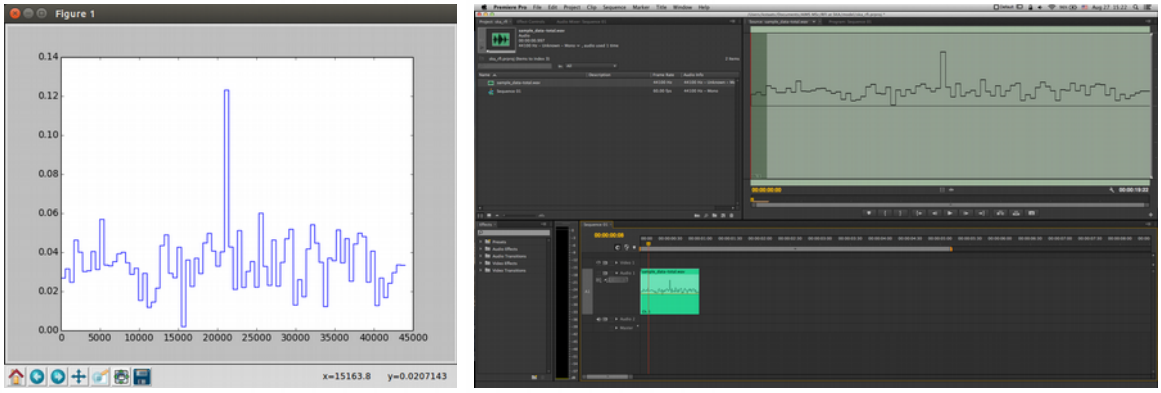
A toy project by Kai Staats

## Overview

This is a toy project designed to explore the potential of using audio editing software as a means of filtering data. While not likely an end-game, this may give us a better understanding of some of the means by which we may apply signal processing algorithms to the ultimate goal of machine learning automation.

## Instructions

1. Place the script 'data2wav.py' and the sample data file in the same directory.
2. $ python data2wav.py /[path]/[filename]
3. Start simple, with the defaults. This will generate a 10 second, 44.1KHz .wav file and associated plot for the first channel with all 1000 data points, each of which is associated with a radio *image*. As the original data is recorded at 1Hz (with the sample data set), the script auto-inserts additional copies of the data points (stems) such that the 44,100 Hz is a mathematically correct representation of the original data, no matter the total .wav duration chosen.
4. If you select to append the channels, each will be laid down sequentially, one after the other, such at channel 1 is followed by channel 2, etc. If you select to stack the channels, each will be added to the prior such that all channels will play at the same time in the final audio file. Different ways of viewing the data.
5. You may then open this .wav file in an audio editing app such as Adobe Premiere or Audition, and have fun :)

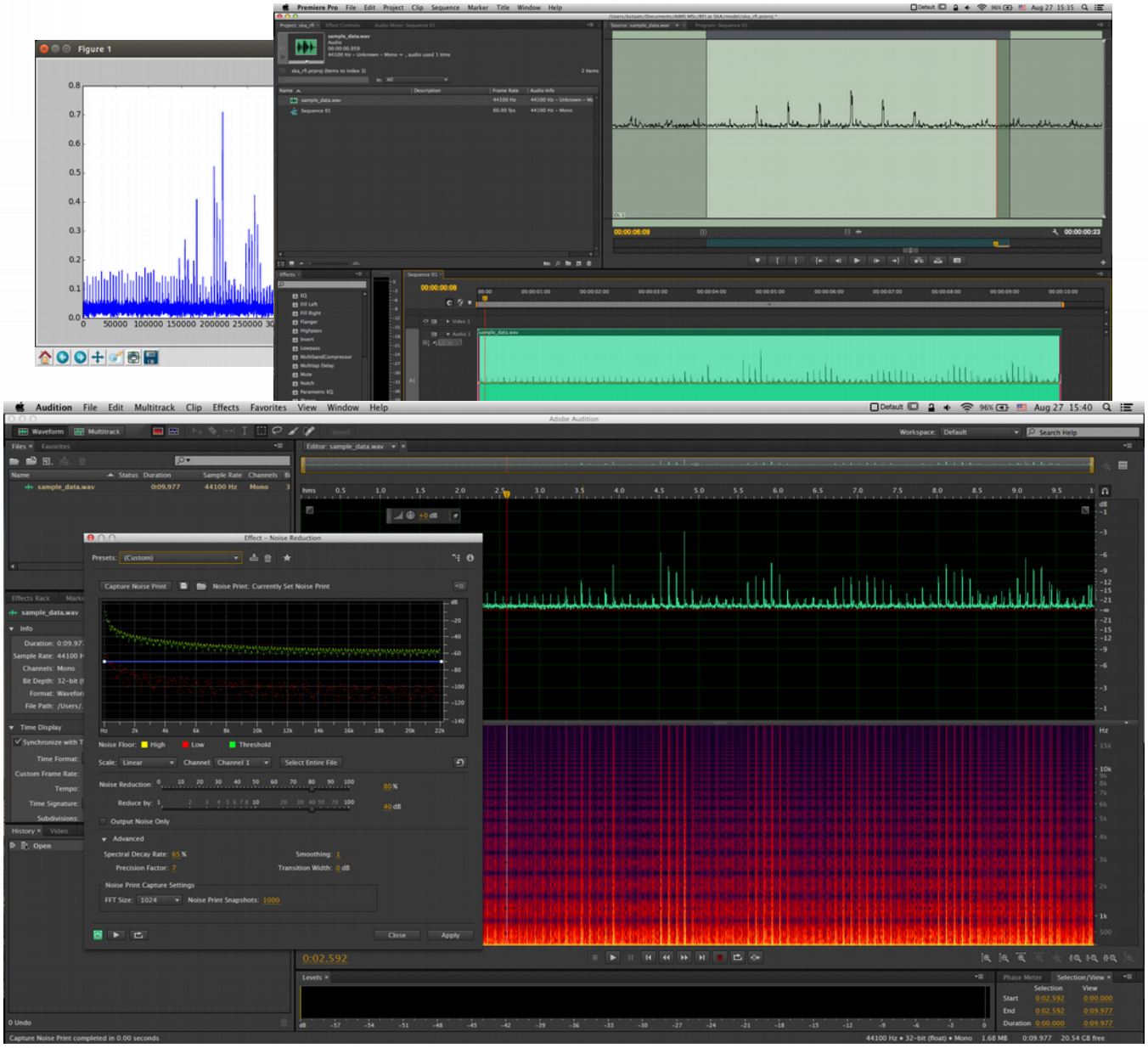



*1337360375_bl_0-1/flag_eg1_pol0.npy, ch0, 1000 data points, 44.1Khz, 1 sec duration*

6. Run the script again, this time selecting 2 or more channels. Each channel will be appended to the prior, creating a longer .wav file. As such, each channel is played back-to-back. The repeating patterns of the signal will be audibly and visibly present across the duration of the playback.

   **CAUTION**: *this script crashed the ipython environment when I entered all 80 channels and attempted to generate a 500 second .wav file. However, 10 seconds works well, even if the data is a bit crowded.*

*(see screenshots on following page)*




*1337360375_bl_0-1/flag_eg1_pol0.npy, ch0-99, 8000 data points, 44.1Khz, 10 sec duration*

7. In audio editing software, you can run audio filters to modify the pitch (non-destructive) or a filter (destructive), or as with AOFLagger, remove specific areas from the waterfall plot.
8. Save the new .wav file, and with the audio2data.py script, you can bring this modified audio file back into the data domain.