Introduction

This project archives the results for the following, in-submission publication:

Inducing musical-interval learning by combining task practice with periods of stimulus exposure alone

Abstract

A key component of musical proficiency is the ability to discriminate between and identify musical intervals, fixed ratios between pitches. Acquiring these skills requires training, but little is known about how to best arrange the trials within a training session. To address this issue, learning on a musical-interval comparison task was evaluated for two four-day training regimens that employed equal numbers of stimulus presentations per day. A regimen of continuous practice yielded no learning, but a regimen that combined practice and stimulus exposure alone generated clear improvement. Learning in the practice-plus-exposure regimen was due to the combination of the two experiences, because two control groups who received only either the practice or the exposure from that regimen did not learn. Post-test performance suggested that comparison learning generalized to an untrained stimulus type and an untrained musical-interval identification task. Naïve comparison performance, but not learning, was better for larger pitch-ratio differences and for individuals with more musical experience. The reported benefits of the practice-plus-exposure regimen mirror outcomes for fine-grained discrimination and speech tasks, suggesting a general learning principle is involved. In practical terms, it appears that combining practice and stimulus exposure alone is a particularly effective configuration for improving musical-interval perception.

Installation

Once you have downloaded this project, to setup the development environment will need to install Anaconda Python 2.7. Once you're finished with the installation, run the following commands on the command line (replacing `[project directory]` with the location of the folder containing the downloaded project).

```
cd [project directory]
conda env create -f environment.yml
source activate apmusic_2018_07_24
```

> In windows these command can be run on the command line that is installed with Anaconda. You will also need to replace the last line as follows:
>
> ```
> activate apmusic_2017_12_03_02
> ```

This setup will take some time to run on your machine as various python and R software packages are installed.

> **Troubleshooting**: If there is any issue with creating the above environment, you can attempt to reconstruct it from the `create_environment.sh` script.

Mac OS X - Make sure you have command-line developer tools installed

On Mac OS X, if you plan to re-run any of the python preprocessing steps, you will need to make sure you have the command-line developer tools installed.

Usage

Once installed, to run the analyses or re-create a figure you will need to have an open programming environment ready. If you haven't closed the window you opened during "Installation", you can use that, otherwise, re-open the command line and run the following

```
cd [project directory]
source activate apmusic_2018_07_24
```

> Again, in Windows these command can be run on the command line that is installed with Anaconda and you will need to replace the last line with `activate apmusic_2018_07_24` (no `source`).

Each figure panel from the paper generally has at least two corresponding files. They are either python or R source files, and they re-create the figure pdf and run the corresponding analyses, respectively. These files are located under the `plots` and `analyses` subfolders. There are some figures with auxiliary analyses (e.g. figure 2A has a secondary analysis by foil instead of regimen), which are included as separate scripts. All scripts associated with a figure include that figure name as a prefix of the filename.

To run any of these files, call either `Rscript` or `python` (as appropriate for the file type) from the same command line you used above.

For example, to re-run the analysis corresponding to Figure 1A you can call the following from the command line.

```
python analyses/fig1A.py
```

And to re-create Figure 1A's pdf:

```
Rscript plots/fig1A.R
```

Yet again, Windows will differ: use `\` instead of `/` in your directory names.

Recreating the preprocessed data

The scripts for each figure and analysis load some preprocessed data. All preprocessed data files are listed in `preprocessing/files.txt`. If you want to recreate one of these files you can do so by finding the script with the same name prefix as the file.

For example, the preprocessed data file `preprocessing/data/discrim_2017-12-31.csv` can be recreated (with a new date suffix) by calling

```
Rscript preprocessing/discrim.R
```

All preprocessed files are stored under the `preprocessing/data` subfolder, while the `data` subfolder contains the original data, as collected during the experiment.

WARNING: some of the preprocessing steps can be time consuming to run.

To start all analyses over from scratch, delete all of the preprocessing data files under the `preprocessing/data` directory. Then, rerun the preprocessing scripts in the following order

1. Aggregate discrimination data - `Rscript preprocessing/discrim.R`
2. Aggregate classification data - `Rscript preprocessing/class.R`
3. Multi-level model fitting - `python preprocessing/multi_level_sample.py`
4. Multi-level model fitting with musical experience - `python preprocessing/music_multi_level_samp`
5. Each figure's preprocessing scripts - `python preprocessing/fig1A.py`, then `python preprocessing/fig1B.py`, etc...

Steps 2 and 3 will be the most time consuming (e.g. ~30 minutes on a 2015 MacBook Air).

If you re-process any data you will have to update the file name listed in `preprocessing/files.txt` so that any scripts relying on this data will use the newly created files. To be absolutely certain you are using newly created files you can delete the old scripts (or move them to another folder).

Editing the scripts

You can use a language specific editor:

- For R - RStudio
- For python - Spyder

You can also use any decent code editor to edit both file types (listed in approximate order from fastest to slowest to learn).

- VSCode
- Sublime
- Emacs
- Vim

These generic editors sometimes require you to install specific plugins to read R and python files.

Regimen Abbreviations

The training regimens, as described in the paper have been given abbreviations in the analysis: AP (Practice+Exposure), LA (All-Practice), SA (Practice+Silence),and P (Exposure+Silence).

Comparison to Published Figures and Statistics

The resulting pdfs from the analysis were imported into Adobe Illustrator and the aesthetics modified for improved legibility and consistency across plots. Thus the published figures will differ somewhat from those produced by these scripts: however, the actual data and reported statistics as represented in the final paper should remain *unchanged*. Small variations in model fits and the resulting statistics will occur due variations in MCMC sampling.

Feedback/Help

If you have any trouble reproducing the analysis or installing the necessary software please feel free to contact the first author