VAG tool documentation

Tanut Aranchayanont

September 11, 2016

This is an update summary of VAG signal processing scripts used in the article *Spectral Analysis on Vibroartrographic Signal of Total Knee Arthroplasty*. There are two issues intended to solve in this version:

- Break the file allocation dependency. Former script use file allocation convention to run script properly, *i.e.*, the new data set need to be place in the correct directory, for example, `../../data/`.
- Use R library for plotting. `ggplot` library provides a publication-quality plotting library and is open source (in progress).

The overview of all code and folder structure is shown below:

```
vag-tool
 |__doc // documentation latex source code
 |__lib // external library used to write up the script
 |   |__cdfa
 |   |__gsl-2.2.1
 |   |__libeemd
 |   |__README
 |__src // code written in this project
     |__signal_to_imfs
     |__signal_to_imfs_all
     |__remove_spike
     |__script_dfa.sh
     |__use_dfa
     |__use_dfa_all
     |__...
```

Installation

Prerequisite is scientific python and r which can be easily installed on Debian linux with the command:

```
$ sudo apt-get install build-essential python-pip3 python3 ipython3 r-base
$ pip install numpy scipy matplotlib
```

Then copy the folder `vag-tool` to your computer. In order to make command line know the new script, we have to export the new path to the folder, which can be done easily by:

```
$ export PATH=$PATH:/path/to/vag-tool/src/
```

Additionally, we can add this line to the `.bash_profile` or `.bashrc` to make this line execute every time we attend the terminal session.

Usage

EEMD with libeemd

For IMFs, EEMD algorithm take a single time signal located at path, `in_path` and output the set of IMF to the output directory `out_dir`. In case multiple input files, there is a script called `signal_to_imfs_all` which replace path to input file, `in_path` with path to input directory `in_dir`

```
$ signal_to_imfs in_path out_dir\\
$ signal_to_imfs_all in_dir out_dir
```

DFA

After downloading library DFA and `make` process, there will be an executable `dfa`. This exececutable can be used directly one by one to create DFA pairs but not very convenient when we would like to process over the directory so that `script_dfa.sh` is introduced.

```
$ script\_dfa in_dir dfa_pair_out_dir
```

The parameter α is calculated from DFA pairs. Each IMFs is composed back with the alpha criterion, for example, $\alpha > 0.5$. Both steps (1) obtain α of each IMF (2) composite the IMFs back due to alpha criterion are done with `use_dfa` and `use_dfa_all`.

```
$ use_dfa_all imfs_dir dfa_pair_dir alpha_out_dir final_signal_dir
```

Working example

Let assume there are the data at directory `sample`:

```
sample
├── s1.txt
└── s2.txt
```

then processed with `signal_to_imfs_all`

```
$ signal_to_imfs_all sample/ imfs/
```

and finally out put to the folder `imfs`

```
imfs
└── s1
    ├── 0.txt
    ├── 1.txt
    ├── 2.txt
    └── 3.txt
```

```
|
|
|   |
|   |__4.txt
|__s2
    |__0.txt
    |__1.txt
    |__2.txt
    |__3.txt
    |__4.txt
```