Seispy
User Manual of Receiver Functions

Mijian Xu[1,*]

[1] *School of Earth Science and Engineering, Nanjing University*
[*] *Email: gomijianxu@gmail.com*

March 14, 2016

Contents

1 Introduction

Seispy is a Python package to process Seismic waveform and receiver functions. This package not only provide a python module but also include scripts to batch receiver functions. This package can only support Linux and Mac OSX platform with Python 3.x (I am not sure the compatibility with the Python 3.5.x+). It is distributed under the GNU General Public License Version 3 (GPLv3) as published by the Free Software Foundation (http://www.gnu.org/licenses/gpl.html).

2 Installation

2.1 Dependencies

Seispy depends on standard libraries of Python 3.x and the Obspy module (http://docs.obspy.org). Moreover, This package require GMT 5.x (http://gmt.soest.hawaii.edu) in plotting receiver functions.

2.2 Installation

1. Install Obspy and GMT.
2. Download Seispy. Opening terminal, run the following commands:

```
git clone https://github.com/xumi1993/seispy.git
```

3. Install seispy by running following commands:

```
cd seispy
python setup.py install
```

4. If the scripts are required, please add executable scripts to environment path. The executable scripts include `pickrf.py`, `pickrf_R.py`, `updateCatalog.py` and `makeRF4station.py`, which used to calculate and filter receiver functions in a batch.

3 Tutorial of calculating receiver functions

3.1 A function of iterative deconvolution

Using this `seispy.decov` to calculate receiver functions by iterative deconvolution.

3.1.1 decovit

```
RFI, RMS, IT = seispy.decov.decovit(UIN,WIN,DT,NT,TSHIFT,
                F0,ITMAX,MINDERR)
```

Parameters:

UIN = numerator (radial for PdS in `numpy.ndarray`)

WIN = denominator (vertical component for PdS in `numpy.ndarray`)

DT = sample interval (s)

NT = number of samples

TSHIFT = Time until beginning of receiver function (s)

F0 = width of gaussian filter

ITMAX = max # iterations

MINDERR = Min change in error required for stopping iterations.

Output

RFI = receiver function (`numpy.ndarray`).

RMS = Root mean square error for predicting numerator after each iteration.

IT = number of iterations.

3.1.2 gaussFilter

Design Gauss filter.

```
gauss = seispy.decov.gaussFilter( dt, nft, f0 )
```

Parameters:

dt = sample interval (s)

nft = number of samples

f0 = width of gaussian filter

Output:

gauss = series of gauss filter in `numpy.ndarray`

3.2 Calculating receiver function in a batch

3.2.1 Usage

The `makeRF4station.py` is a script to calculate receiver function in a batch.

```
python makeRF4station.py –Sstation \
       –Yyearmin/monthmin/daymin/yearmax/monthmax/daymax\
       –Cchannel [–T[+E|+N]] [–r] paraRF.cfg
```

3.2.2 Synopsis

-S The station name. It must be same as the directory name, which include original data of SAC format.

-Y A rough date range during this operation.

-C A component name in SAC file name (e.g., *BHZ, HHZ, 1*.)

-T Attach **+E** to switch east component sign. Attach **+N** to switch north component sign.

-r Only calculate radial receiver functions

paraRF.cfg Parameters of calculating receiver functions (see Table 3.1).

3.2.3 Example

```
python makeRF4station.py –SNJ2 –Y2015/1/1/2015/12/1 –CBHZ paraRF.cfg
python makeRF4station.py –SNJ2 –Y2015/1/1/2015/12/1 –CBHE –T+E paraRF.c
```

Note:

1. This program will calculate P-wave arrival time automatically. Therefore, locations of station must be written in SAC header value.

2. The `tolerance` value in seconds will define the time window within which the program will try to associate a seismic file to an event file, by using either its name or the information contained in the header. It is up to the user to find the best compromise: a value too small will let orphans and a value too large will bring confusion since several files could be associated to a seismic event (*See Splitlab User Manual*).

Table 3.1: Options in config file

Option	Function
`data_path`	Directory of original data
`out_path`	Data after cutting by time window
`RF_path`	Directory of receiver functions
`image_path`	Directory of Figures
`evt_list`	Directory of Catalog
`gate_mw`	Low boundary of magnitude
`gate_dis1`	Low boundary of epicenteral distance
`gate_dis2`	High boundary of epicenteral distance
`time_before`	Low boundary of time window
`time_after`	High boundary of time window
`tolerance`	This value in seconds will define the time window within which the program will try to associate a seismic file to an event file.
`offset`	A time duration between the event time and the starting time of your seismograms.
`gate_noise`	A threshold value in SNR analysis.
`gauss`	Gauss factor.
`freqmin`	Low cut-off frequency of a filter before calculating receiver function.
`freqmax`	High cut-off frequency of a filter before calculating receiver function.
`sampling`	time interval in second, which the program will resample waveforms to this sampling rate.

3. The `offset` is the time duration between the event time and the starting time of your seismograms. Ideally, this offset should be identical to the "request start time" defined in the previous window but the data management center may have sent you data beginning later than requested. The offset value represents this difference (*See Splitlab User Manual*).

3.3 Filter receiver functions manually

`pickrf.py/pickrf_R.py` are GUI scripts to process receiver functions after running `makeRF4station.py`

3.3.1 Usage

```
python pickrf.py –Sstation paraRF.cfg
```

3.3.2 Synopsis

All of these arguments are same as that in `makeRF4station.py`.

3.3.3 Interface

Figure 3.1 shows this GUI interface

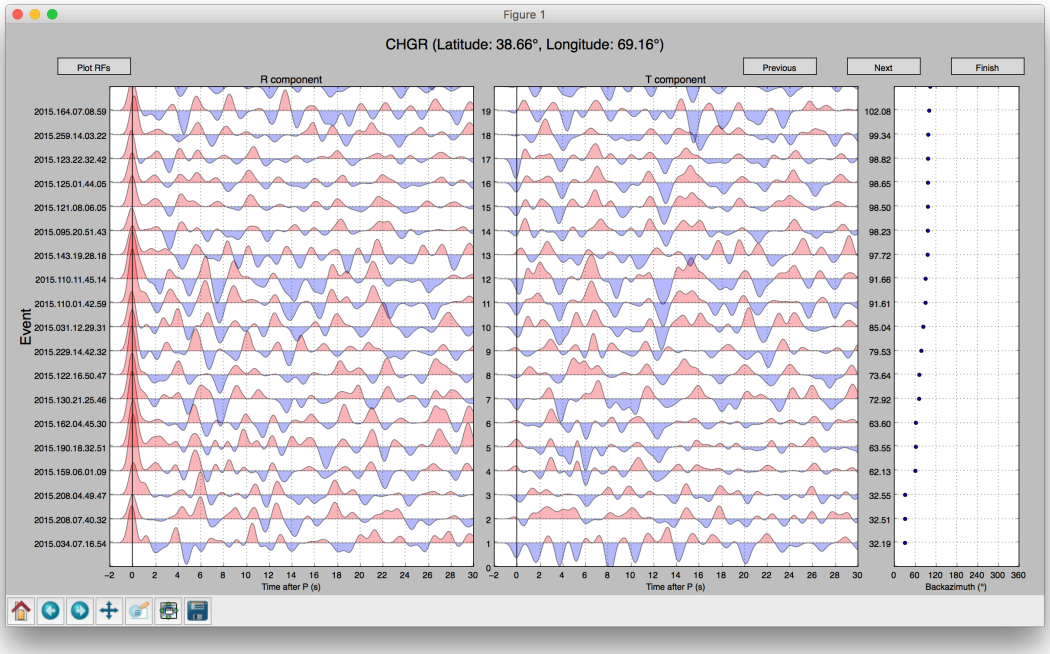

Figure 3.1: GUI Interface to process receiver functions.