AIMBAT
User Manual
Version 0.1.1

Xiaoting Lou[1*]

[1]*Department of Earth and Planetary Sciences, Northwestern University, Evanston, Illinois, USA*

[*]*Email: xlou@u.northwestern.edu*

October 1, 2012

Contents

1 Introduction

AIMBAT (Automated and Interactive Measurement of Body wave Arrival Times) is an open-source software package for efficiently measuring teleseismic body wave arrival times for large seismic arrays (*Lou et al.*, 2012). It is based on a widely used method called MCCC (Multi-Channel Cross-Correlation) developed by *VanDecar and Crosson* (1990). The package is automated in the sense of initially aligning seismograms for MCCC which is achieved by an ICCS (Iterative Cross Correlation and Stack) algorithm. Meanwhile, a GUI (graphical user interface) is built to perform seismogram quality control interactively. Therefore, user processing time is reduced while valuable input from a user's expertise is retained. As a byproduct, SAC (*Goldstein et al.*, 2003) plotting and phase picking functionalities are replicated and enhanced.

Modules and scripts included in the AIMBAT package were developed using Python programming language (`http://www.python.org`) and its open-source modules on the Mac OS X platform since 2009. The original MCCC code (*VanDecar and Crosson*, 1990) was transcribed into Python. The GUI of AIMBAT was inspired and initiated at the 2009 EarthScope USArray Data Processing and Analysis Short Course (`http://www.iris.edu/hq/es_course/`). AIMBAT runs on Mac OS X, Linux/Unix and Windows thanks to the platform-independent feature of Python. It's been tested on Mac OS 10.6.8 and 10.7 and Fedora 16.

The AIMBAT software package 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 Install Dependencies

The AIMBAT package depends on the standard libraries of Python 2.7 (http://docs.python.org/library/), including `os`, `sys`, `ConfigParser(argparse)`, `optparse`, `contextlib`, `pickle/cPickle`, `gzip`, and `bz2`. AIMBAT also utilizes Numpy (http://numpy.scipy.org) and Scipy (http://scipy.org) for numerical array computation, and Matplotlib (*Hunter*, 2007) for 2-D plotting and GUI applications. These packages need to be installed before using AIMBAT.

2.1.1 Mac

For Mac users, we recommend Macports (http://http://www.macports.org/) to install the dependent packages of AIMBAT. After installing Macports, run the following commands with superuser privilege:

```
port install python27
port install py27-numpy
port install py27-scipy
port install py27-matplotlib
port install py27-ipython
port install python_select
```

Python 2.7 is thus installed to the `<prefix>` directory, which is `/opt/local/Library/Frameworks/Python.framework/Versions/2.7` for this case. Corresponding packages Numpy, Scipy and Matplotlib are installed to the global site-packages directory `<prefix>/lib/python2.7/site-packages`.

Installation of the last two packages are optional. `ipython` is an enhanced interactive Python shell. `python_select` is used to select default Python version by the following command:

```
port select --set python python27
```

2.1.2 Linux

For Linux users, package management tools work similarly, such as `yum` on Red Hat/Fedora system:

```
yum install python.x86_64
yum install numpy.x86_64
yum install scipy.x86_64
yum install python-matplotlib.x86_64
```

and `aptitude` on Debian system:

```
aptitude install python
aptitude install python-numpy
aptitude install python-scipy
aptitude install python-matplotlib
```

The `<prefix>` directory is `/usr`.

2.2 Install pysmo.sac and pysmo.aimbat

AIMBAT is released as a sub-package of pysmo in the name of `pysmo.aimbat` along with another sub-package `pysmo.sac`. The latest releases of `pysmo.sac` and `pysmo.aimbat` are available for download at both

```
http://www.earth.northwestern.edu/~xlou/aimbat.html
```

and github:

```
https://github.com/pysmo/sac
https://github.com/pysmo/aimbat
```

Decompress the gzipped source tar balls in the directory where you want to install the packages (`<pkg-install-dir>`):

```
tar zxvf pysmo-sac-0.5.tar.gz
tar zxvf pysmo-aimbat-0.1.1.tar.gz
```

2.2.1 Install pysmo.sac

Python module `Distutils` is used to write a `setup.py` script to easily build, distribute, and install `pysmo.sac`. In the directory `<pkg-install-dir>/pysmo-sac-0.5`, type

```
python setup.py build
python setup.py install
```

to install it and its package information file `pysmo.sac-0.5-py2.7.egg-info` to the global site-packages directory `<prefix>/lib/python2.7/site-packages`, which is the same as Numpy, Scipy, and Matplotlib.

If you don't have write permission to the global site-packages directory, use the "`--user`" option to install to `<userbase>/lib/python2.7/site-packages`:

```
python setup.py install --user
```

2.2.2 Install pysmo.aimbat

Three sub-directories are included in the `<pkg-install-dir>/pysmo-aimbat-0.1.1` directory: `example`, `scripts` and `src`, which contain example SAC data files, Python scripts to run at command line, and Python modules to install, respectively.

The core cross-correlation functions in `pysmo.aimbat` are written in both Python/Numpy (`xcorr.py`) and Fortran (`xcorr.f90`). Therefore, we need to use Numpy's `Distutils` module for enhanced support of Fortran extension. The usage is similar to the standard `Distutils`.

In the directory `<pkg-install-dir>/pysmo-aimbat-0.1.1`, type

```
python setup.py build --fcompiler=gfortran
python setup.py install
```

to install the `src` directory to `<prefix>/lib/python2.7/site-packages/pysmo/aimbat`. Other Fortran compilers can be specified instead of gfortran.

Add `<pkg-install-dir>/pysmo-aimbat-0.1.1/scripts` to environment variable `PATH` in a shell's start-up file for command line execution of the scripts. Typically for Bash and C shell users, type

```
export PATH=$PATH:<pkg-install-dir>/pysmo-aimbat-0.1.1/scripts
setenv PATH $PATH:<pkg-install-dir>/pysmo-aimbat-0.1.1/scripts
```

in `.bashrc` and `.cshrc` files, respectively.

To test the installation of the `pysmo.sac` and `pysmo.aimbat` packages, type

```
from pysmo import sac
from pysmo import aimbat
```

in a Python shell.

3 Tutorial

3.1 Parameter Configuration

Matplotlib works with six GUI (Graphical User Interface) toolkits: WX, Tk, Qt(4), GTK, Fltk and macosx (`http://matplotlib.org/contents.html`). The GUI of AIMBAT utilizes GUI neutral widgets and GUI neutral event handling API (Application Programming Interface) to support interactive plotting (`http://matplotlib.org/api/widgets_api.html`, `http://matplotlib.org/users/event_handling.html`). Examples given in this manual are using the default toolkit `Tk` and backend `TkAgg`. See `http://matplotlib.org/faq/usage_faq.html#what-is-a-backend` and `http://matplotlib.org/users/customizing.html#customizing-matplotlib` for explanation of the backend and how to customize it. In short, put the following line in your `matplotlibrc` file:

```
backend :  TkAgg #Agg rendering to a Tk canvas
```

Other parameters for the package can be set up by a configuration file `ttdefaults.conf`, which is interpreted by the module `ConfigParser`. This configuration file is searched in the following order:

(1) file `ttdefaults.conf` in the current working directory

(2) file `.aimbat/ttdefaults.conf` in your `HOME` directory

(3) a file specified by environment variable `TTCONFIG`

(4) file `ttdefaults.conf` in the directory where AIMBAT is installed

An example of the `ttdefaults.conf` file and its explanations are given in Table 1.

Python scripts in the `<pkg-install-dir>/pysmo-aimbat-0.1.1/scripts` can be executed from the command line. The command line arguments are parsed by the `optparse` module to improve the scripts' flexibility. If conflicts existed, the command line options override the default parameters given in the configuration file `ttdefaults.conf`. Run the scripts with the "-h" option for the usage messages.

File `ttdefaults.conf`	Description
[sacplot]	
colorwave = blue	Color of waveform
colorwavedel = gray	Color of waveform which is deselected
colortwfill = green	Color of time window fill
colortwsele = red	Color of time window selection
alphatwfill = 0.2	Transparency of time window fill
alphatwsele = 0.6	Transparency of time window selection
npick = 6	Number of time picks (plot picks: t0-t5)
pickcolors = kmrcgyb	Colors of time picks
pickstyles = – :	Line styles of time picks (use second one if ran out of color)
figsize = 8 10	Figure size for `plotphase.py`
rectseis = 0.1 0.06 0.76 0.9	Axes rectangle size within the figure
minspan = 5	Minimum sample points for SpanSelector to select time window
srate = -1	Sample rate for loading SAC data. Read from first file if srate < 0
[sachdrs]	
twhdrs = user8 user9	SAC headers for time window beginning and ending
ichdrs = t0 t1 t2	SAC headers for ICCS time picks
mchdrs = t2 t3	SAC headers for MCCC input and output time picks
hdrsel = kuser0	SAC header for seismogram selection status
qfactors = ccc snr coh	Quality factors: cross-correlation coefficient, signal-to-noise ratio, time domain coherence
qheaders = user0 user1 user2	SAC Headers for quality factors
qweights = 0.3333 0.3333 0.3333	Weights for quality factors
[iccs]	
srate = -1	Sample rate for loading SAC data. Read from first file if srate < 0
xcorr_modu = xcorrf90	Module for calculating cross-correlation: xcorr for Numpy or xcorrf90 for Fortran
xcorr_func = xcorr_fast	Function for calculating cross-correlation
shift = 10	Sample shift for running coarse cross-correlation
maxiter = 10	Maximum number of iteration
convepsi = 0.001	Convergence criterion: epsilon
convtype = coef	Type of convergence criterion: coef for correlation coefficient, or resi for residual
stackwgt = coef	Weight each trace when calculating array stack
fstack = fstack.sac	SAC file name for the array stack
[mccc]	
srate = -1	Sample rate for loading SAC data. Read from first file if srate < 0
ofilename = mc	Output file name of MCCC. Use ”$evdate.mc$phase” if mc
xcorr_modu = xcorrf90	Module for calculating cross-correlation: xcorr for Numpy or xcorrf90 for Fortran
xcorr_func = xcorr_faster	Function for calculating cross-correlation
shift = 10	Sample shift for running coarse cross-correlation
extraweight = 1000.	Weight for the zero-mean equation in MCCC weighted lsqr solution
lsqr = nowe	Type of lsqr solution: no weight
#lsqr = lnco	Type of lsqr solution: weighted by correlation coefficient, solved by lapack
#lsqr = lnre	Type of lsqr solution: weighted by residual, solved by lapack
rcfile = .mcccrc	Configuration file for MCCC parameters (deprecated)
evlist = event.list	File for event hypocenter and origin time (deprecated)
[signal]	
tapertype = hanning	Taper type
taperwidth = 0.1	Taper width

Table 1: Example of AIMBAT configuration file `ttdefaults.conf`.

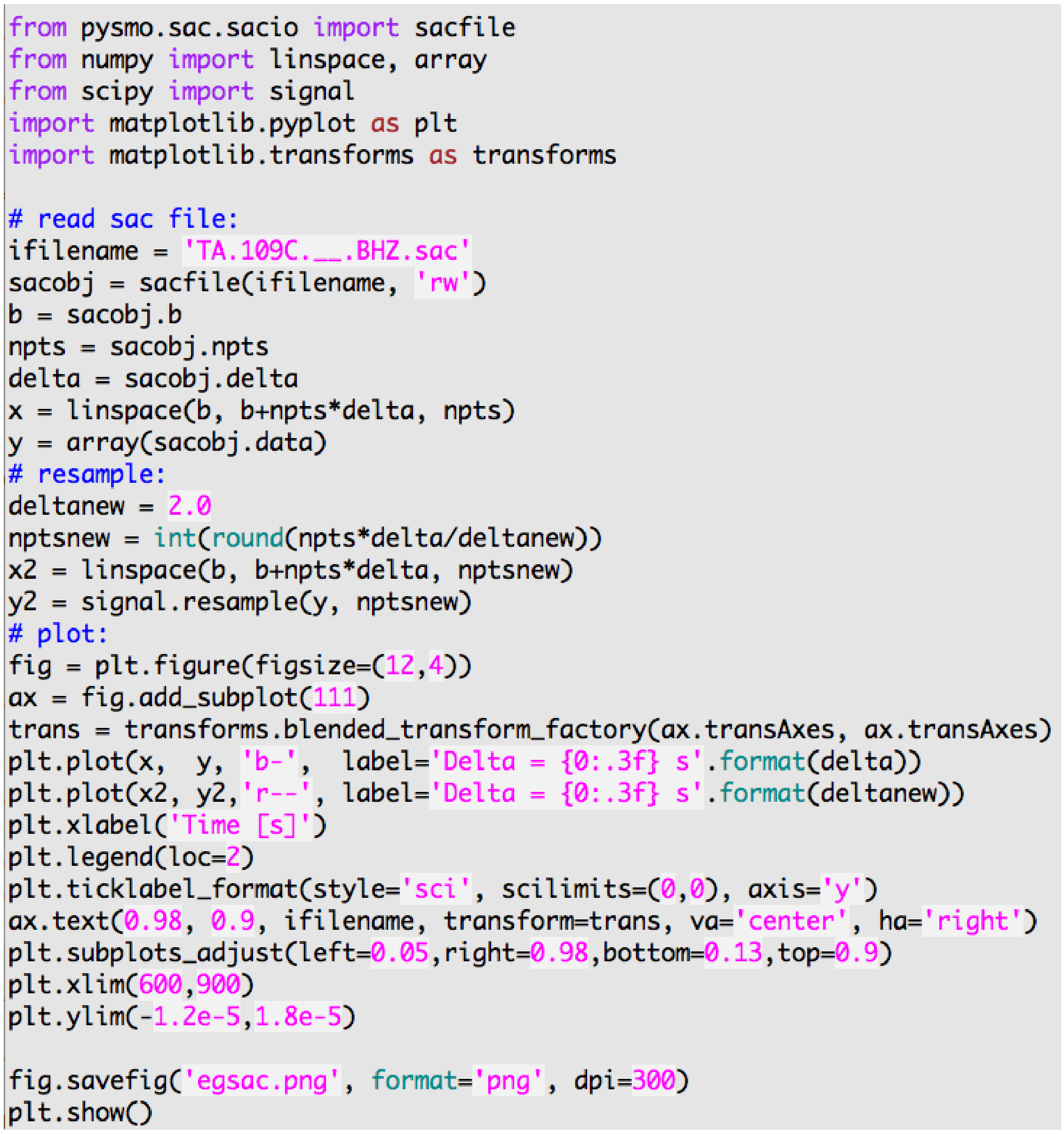
3.2 SAC Data Access

3.2.1 Python Object for SAC Files

The `pysmo.sac` package is developed to read and write individual SAC files. The Python class `sacfile` of module `sacio` opens a SAC file and returns an object including data and all

SAC header variables as its attributes. Modifications of object attributes are saved to file. It is written purely in Python so that it also runs with Jython (`http://www.jython.org`).

The `<pkg-install-dir>/pysmo-aimbat-0.1.1/scripts/egsac.py` script (Figure 1) gives a simple example to read, resample and plot a seismogram using pysmo, Scipy and Matplotlib. You can type the codes in a Python/iPython shell, or run as a script in the directory `<pkg-install-dir>/pysmo-aimbat-0.1.1/example/Event_2011.09.15.19.31.04.080` (referred to as `<example-event-dir>` hereafter).



```
from pysmo.sac.sacio import sacfile
from numpy import linspace, array
from scipy import signal
import matplotlib.pyplot as plt
import matplotlib.transforms as transforms

# read sac file:
ifilename = 'TA.109C.__.BHZ.sac'
sacobj = sacfile(ifilename, 'rw')
b = sacobj.b
npts = sacobj.npts
delta = sacobj.delta
x = linspace(b, b+npts*delta, npts)
y = array(sacobj.data)
# resample:
deltanew = 2.0
nptsnew = int(round(npts*delta/deltanew))
x2 = linspace(b, b+npts*delta, nptsnew)
y2 = signal.resample(y, nptsnew)
# plot:
fig = plt.figure(figsize=(12,4))
ax = fig.add_subplot(111)
trans = transforms.blended_transform_factory(ax.transAxes, ax.transAxes)
plt.plot(x,  y, 'b-',  label='Delta = {0:.3f} s'.format(delta))
plt.plot(x2, y2,'r--', label='Delta = {0:.3f} s'.format(deltanew))
plt.xlabel('Time [s]')
plt.legend(loc=2)
plt.ticklabel_format(style='sci', scilimits=(0,0), axis='y')
ax.text(0.98, 0.9, ifilename, transform=trans, va='center', ha='right')
plt.subplots_adjust(left=0.05,right=0.98,bottom=0.13,top=0.9)
plt.xlim(600,900)
plt.ylim(-1.2e-5,1.8e-5)

fig.savefig('egsac.png', format='png', dpi=300)
plt.show()
```

Figure 1: The `<pkg-install-dir>/pysmo-aimbat-0.1.1/scripts/egsac.py` script which produces Figure 2.

In this example, a SAC file named `TA.109C.__.BHZ.sac` is read in as a sacfile object. The time array is calculated from SAC headers. The data array is resampled from interval 0.025 to 2.0 seconds using Scipy's signal processing module. Result is displayed in Figure 2.

Add the following codes to write the resampled seismogram to file `TA.109C.__.BHZ.sac`:

```
sacobj.delta = deltanew
sacobj.npts = nptsnew
sacobj.data = y2
```

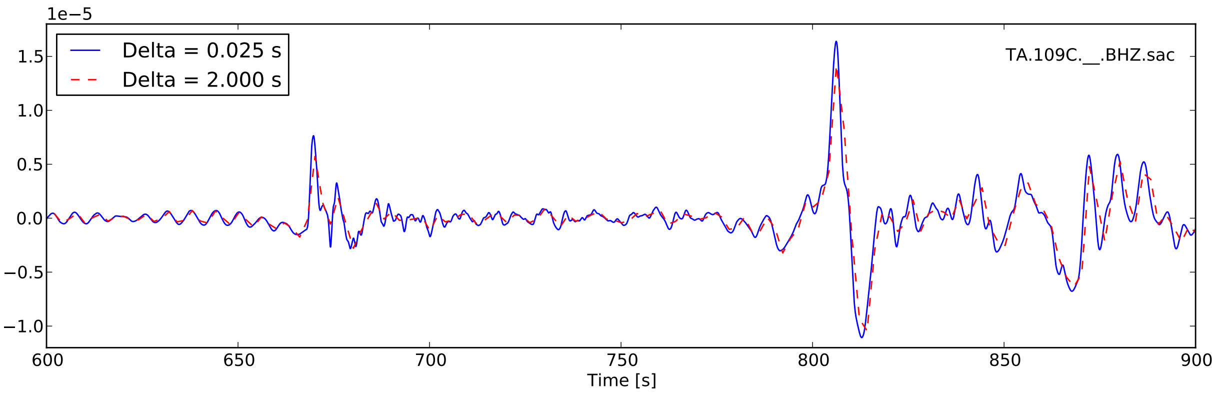


Figure 2: Example of reading, resampling and plotting a SAC file.

3.2.2 Python Pickle for SAC Files

The `pysmo.sacio` module converts SAC files to `sacfile` objects. Any modification of the objects are instantly written to files. In data processing, the user may want to abandon changes made earlier, which brings the need of a buffer for the `sacfile` objects. The `SacDataHdrs` class in the `pysmo.aimbat.sacpickle` module is written on top of `pysmo.sacio` to serves this purpose by reading a SAC file and returning a `sacdh` object that is very similar of the `sacfile` object. Essentially, the `sacdh` object is a copy of the the `sacfile` object in the memory, except that SAC headers 't0-t9', 'user0-user9', 'kuser0-kuser2' are saved in three Python lists. A `gsac` object of the `SacGroup` class consists of a group of `sacdh` objects from event-based SAC data files, earthquake hypocenter information and station locations. An additional step is required to save changes in the `gsac` object to files.

In order to avoid frequent SAC file I/O, the `pickle/cPickle` module is used for serializing and de-serializing the `gsac` object structure. Thus the data processing efficiency is improved because reading and writing of SAC files are done only once each before and after data processing. Script sac2pkl.py does the conversions between SAC files and Python pickles. Its usage message can be printed out by running "sac2pkl.py -h" at command line and the result is displayed in Figure 3. For example, in the data example directory <example-event-dir>, run

```
sac2pkl.py -s *Z -o 20110915.19310408.bhz.pkl -d 0.025
```

to read 163 vertical component seismograms at a sample interval of 0.025 s and convert to a gsac object which is saved in the pickle file 20110915.19310408.bhz.pkl.

To save disk space, compressed pickle files in gz and bz2 formats can be generated by:

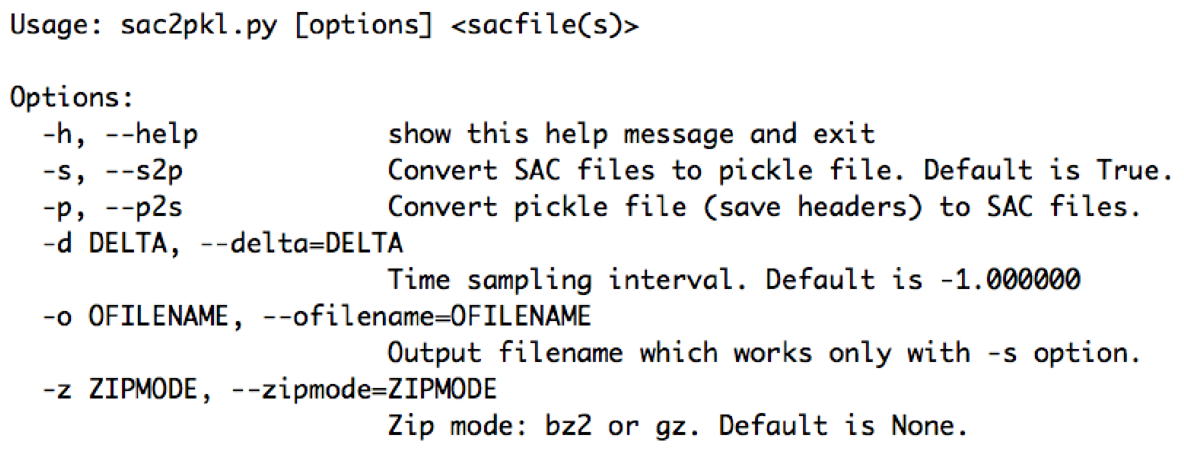
```
sac2pkl.py -s *Z -o 20110915.19310408.bhz.pkl -d 0.025 -z gz
sac2pkl.py -s *Z -o 20110915.19310408.bhz.pkl -d 0.025 -z bz2
```

at the cost of more CPU time.

After processing, run

```
sac2pkl.py 20110915.19310408.bhz.pkl -p
```

to convert the pickle file to SAC files.



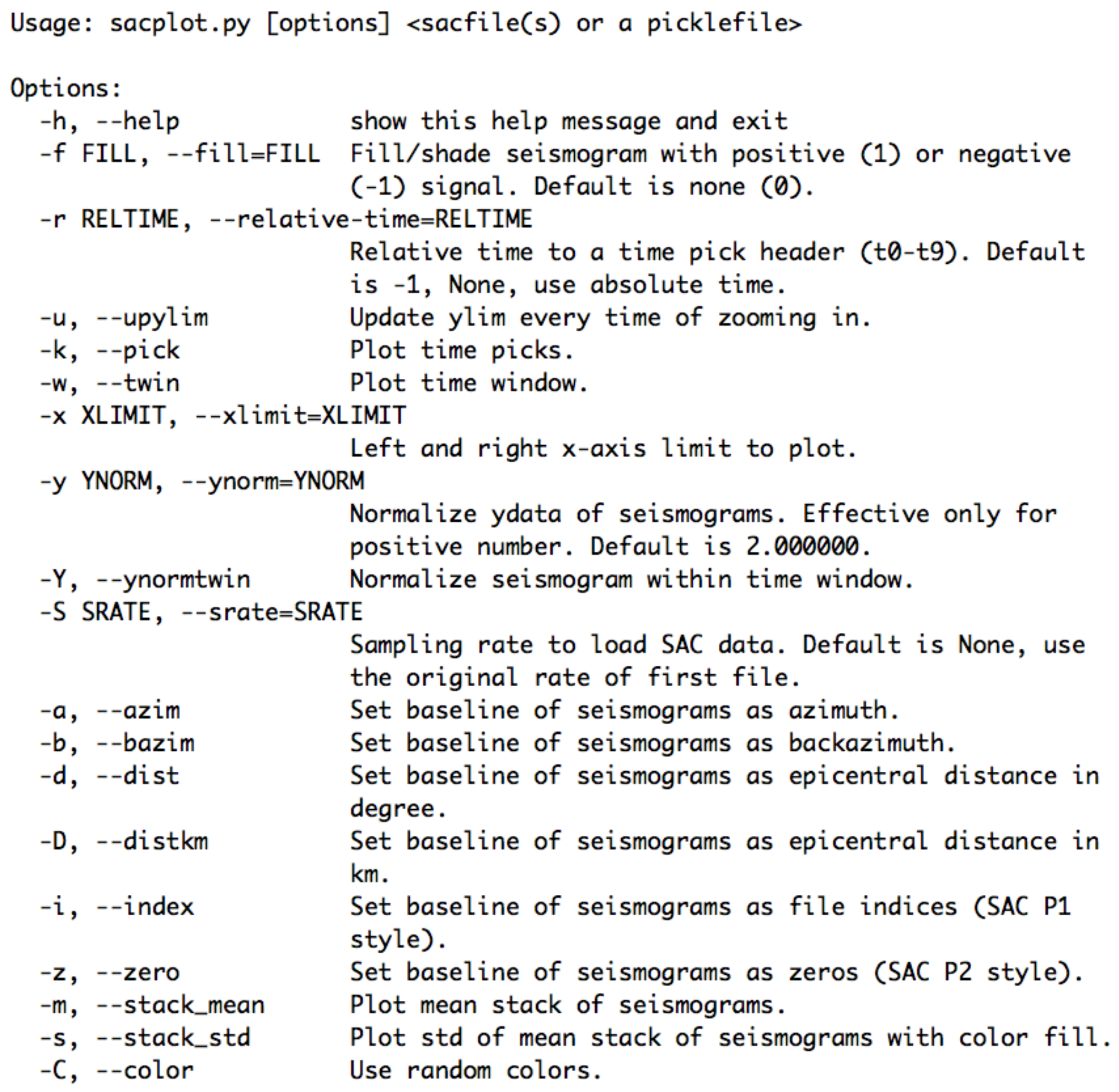
```
Usage: sac2pkl.py [options] <sacfile(s)>

Options:
  -h, --help            show this help message and exit
  -s, --s2p             Convert SAC files to pickle file. Default is True.
  -p, --p2s             Convert pickle file (save headers) to SAC files.
  -d DELTA, --delta=DELTA
                        Time sampling interval. Default is -1.000000
  -o OFILENAME, --ofilename=OFILENAME
                        Output filename which works only with -s option.
  -z ZIPMODE, --zipmode=ZIPMODE
                        Zip mode: bz2 or gz. Default is None.
```

Figure 3: Help message of the sac2pkl.py script.

See the doc string of pysmo.aimbat.sacpickle (type "from pysmo.aimbat import sacpickle; print sacpickle.__doc__" in a Python shell) and http://docs.python.org/library/pickle.html for more information about the Python data structure, pickling and unpickling.

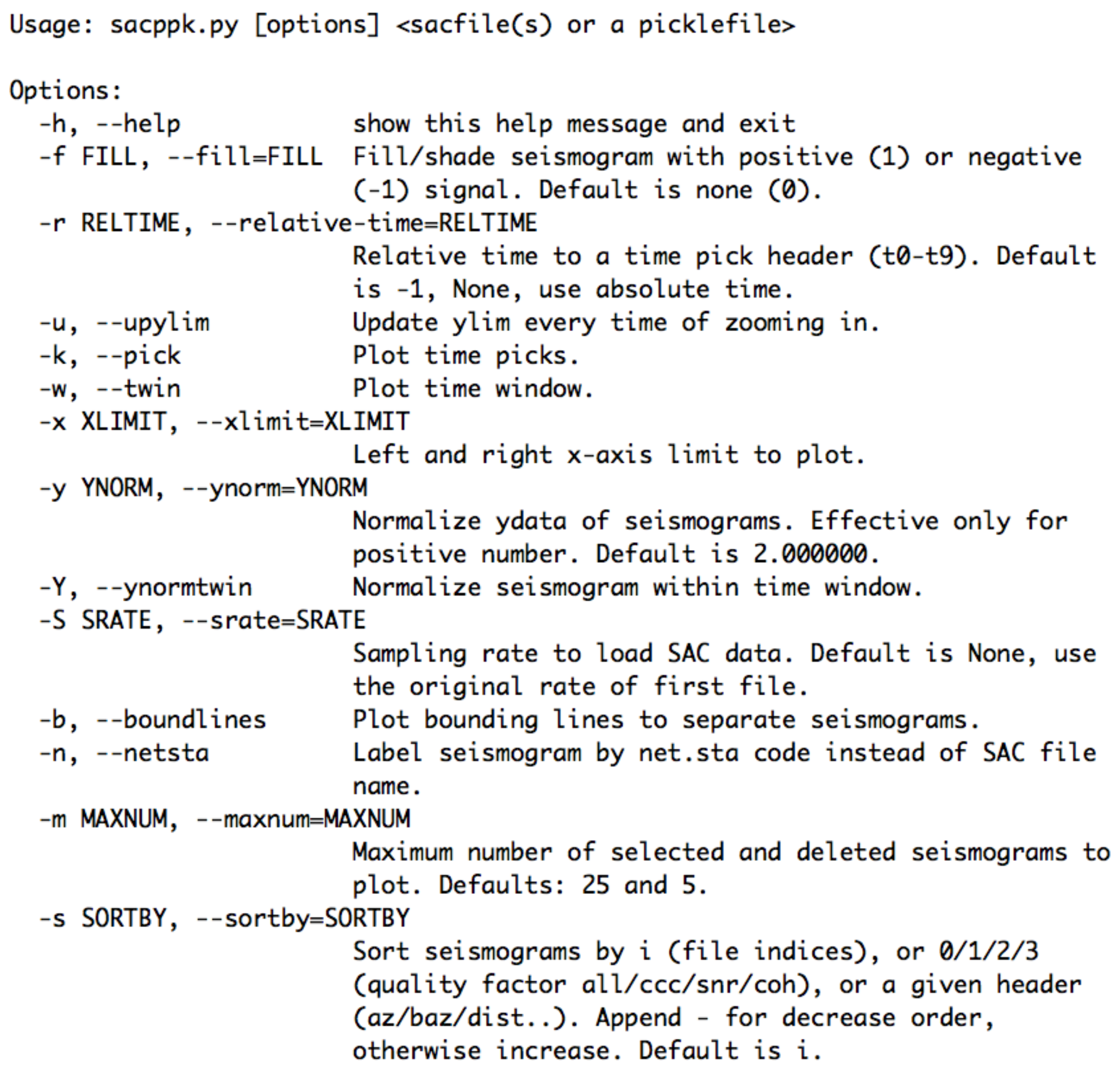
3.3 SAC Plotting and Phase Picking

SAC plotting and phase picking functionalities are replicated and enhanced based on the GUI neutral widgets (such as Button and SpanSelector) and the event (keyboard and mouse events such as `key_press_event` and `mouse_motion_event`) handling API of Matplotlib (*Hunter*, 2007). They are implemented in two modules `pysmo.aimbat.plotphase` and `pysmo.aimbat.pickphase`, which are used by corresponding scripts `sacplot.py` and `sacppk.py` executable at command line. Their help messages are displayed in Figures 4 and 5.



```
Usage: sacplot.py [options] <sacfile(s) or a picklefile>

Options:
  -h, --help            show this help message and exit
  -f FILL, --fill=FILL  Fill/shade seismogram with positive (1) or negative
                        (-1) signal. Default is none (0).
  -r RELTIME, --relative-time=RELTIME
                        Relative time to a time pick header (t0-t9). Default
                        is -1, None, use absolute time.
  -u, --upylim          Update ylim every time of zooming in.
  -k, --pick            Plot time picks.
  -w, --twin            Plot time window.
  -x XLIMIT, --xlimit=XLIMIT
                        Left and right x-axis limit to plot.
  -y YNORM, --ynorm=YNORM
                        Normalize ydata of seismograms. Effective only for
                        positive number. Default is 2.000000.
  -Y, --ynormtwin       Normalize seismogram within time window.
  -S SRATE, --srate=SRATE
                        Sampling rate to load SAC data. Default is None, use
                        the original rate of first file.
  -a, --azim            Set baseline of seismograms as azimuth.
  -b, --bazim           Set baseline of seismograms as backazimuth.
  -d, --dist            Set baseline of seismograms as epicentral distance in
                        degree.
  -D, --distkm          Set baseline of seismograms as epicentral distance in
                        km.
  -i, --index           Set baseline of seismograms as file indices (SAC P1
                        style).
  -z, --zero            Set baseline of seismograms as zeros (SAC P2 style).
  -m, --stack_mean      Plot mean stack of seismograms.
  -s, --stack_std       Plot std of mean stack of seismograms with color fill.
  -C, --color           Use random colors.
```

Figure 4: Help message of the `sacplot.py` script.



```
Usage: sacppk.py [options] <sacfile(s) or a picklefile>

Options:
  -h, --help            show this help message and exit
  -f FILL, --fill=FILL  Fill/shade seismogram with positive (1) or negative
                        (-1) signal. Default is none (0).
  -r RELTIME, --relative-time=RELTIME
                        Relative time to a time pick header (t0-t9). Default
                        is -1, None, use absolute time.
  -u, --upylim          Update ylim every time of zooming in.
  -k, --pick            Plot time picks.
  -w, --twin            Plot time window.
  -x XLIMIT, --xlimit=XLIMIT
                        Left and right x-axis limit to plot.
  -y YNORM, --ynorm=YNORM
                        Normalize ydata of seismograms. Effective only for
                        positive number. Default is 2.000000.
  -Y, --ynormtwin       Normalize seismogram within time window.
  -S SRATE, --srate=SRATE
                        Sampling rate to load SAC data. Default is None, use
                        the original rate of first file.
  -b, --boundlines      Plot bounding lines to separate seismograms.
  -n, --netsta          Label seismogram by net.sta code instead of SAC file
                        name.
  -m MAXNUM, --maxnum=MAXNUM
                        Maximum number of selected and deleted seismograms to
                        plot. Defaults: 25 and 5.
  -s SORTBY, --sortby=SORTBY
                        Sort seismograms by i (file indices), or 0/1/2/3
                        (quality factor all/ccc/snr/coh), or a given header
                        (az/baz/dist..). Append - for decrease order,
                        otherwise increase. Default is i.
```

Figure 5: Help message of the `sacppk.py` script.

3.3.1 SAC Plotting

Options ”-i, -z, -d, -a, and -b” of `sacplot.py` set the seismogram plotting baseline as file index, zero, epicentral distance in degrees, azimuth, and back-azimuth, respectively. The user can run `sacplot.py` directly with the options, or run individual scripts `sacp1.py`, `sacp2.py`, `sacprs.py`, `sacpaz.py`, and `sacpbaz.py` which preset the baseline options and plot seismograms in SAC p1 style, p2 style, record section, and relative to azimuth and back-azimuth. The following commands are equivalent:

`sacplot.py -i` $\Longleftrightarrow$ `sacp1.py` `sacplot.py -z` $\Longleftrightarrow$ `sacp2.py`

`sacplot.py -d` $\Longleftrightarrow$ `sacprs.py`

`sacplot.py -a` $\Longleftrightarrow$ `sacpaz.py`

`sacplot.py -b` $\Longleftrightarrow$ `sacpbaz.py`

```python
from pylab import *
import matplotlib.transforms as transforms
from pysmo.aimbat.sacpickle import loadData
from pysmo.aimbat.plotphase import getDataOpts, PPConfig, sacp1, sacp2, sacprs

# figure axes
fig = figure(figsize=(9,12))
rectp2 = [.09, .050, .8, .15]
rectp1 = [.09, .245, .8, .33]
rectp0 = [.09, .620, .8, .36]
axp2 = fig.add_axes(rectp2)
axp1 = fig.add_axes(rectp1)
axp0 = fig.add_axes(rectp0)

# read data and plot
gsac, opts = getDataOpts()
# prs
opts.ynorm = .95
saclist = gsac.saclist
prs = sacprs(saclist, opts, axp0)
# p1
opts.ynorm = 1.7
p1 = sacp1(saclist, opts, axp1)
# p2
opts.reltime = 0
p2 = sacp2(saclist, opts, axp2)
# set x limits
axp0.set_xlim(625, 762)
axp1.set_xlim(625, 762)
axp2.set_xlim(-45, 65)
# numbering
axs = [axp0, axp1, axp2]
labs = 'ABC'
for ax, lab in  zip(axs, labs):
    tt = '(' + lab + ')'
    trans = transforms.blended_transform_factory(ax.transAxes, ax.transAxes)
    ax.text(-.05, 1, tt, transform=trans, va='center', ha='right', size=16)

fig.savefig('egplot.png', format='png', dpi=300)
show()
```

Figure 6: The `<pkg-install-dir>/pysmo-aimbat-0.1.1/scripts/egplot.py` script which produces Figure 7.

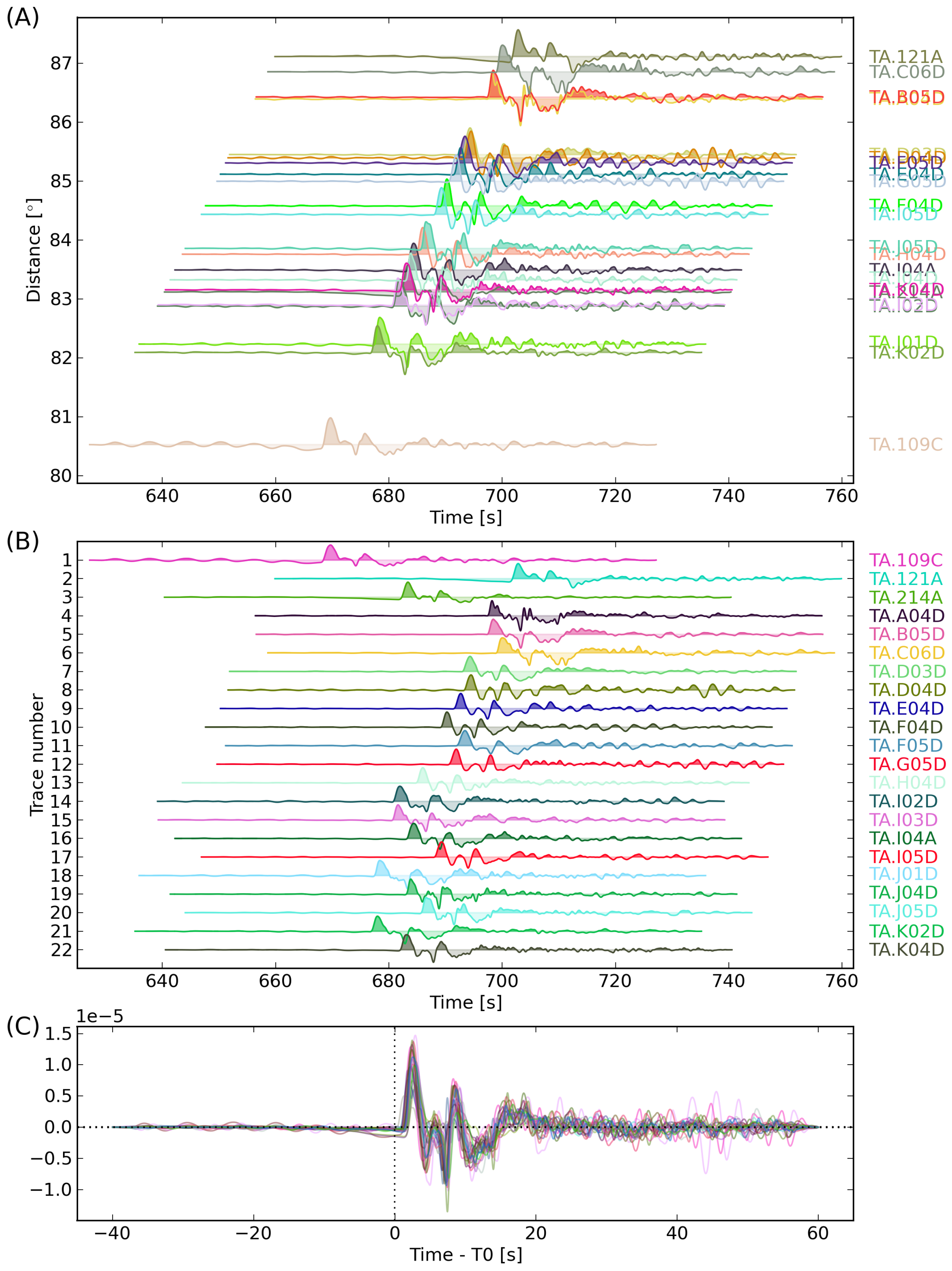


Figure 7: Example of plotting multiple SAC files in (A) record section; (B) SAC p1 style; and (C) SAC p2 style. This figure is the random-color version of Figure 1 in *Lou et al.* (2012).

Input data files need to be supplied to the scripts in the form of either a list of SAC files or a pickle file which includes multiple SAC files. For example, a ”bhz.pkl” file is generated from 22 vertical component seismograms ”TA.[1-K]*Z” by running:

```
sac2pkl.py TA.[1-K]*BHZ -o bhz.pkl -d0.025
```

in the data example directory `<example-event-dir>`. Then the two commands are equivalent:

`sacp1.py TA.[1-K]*Z` $\Longleftrightarrow$ `sacp1.py bhz.pkl`

For large number of seismograms, the pickle file is suggested because of faster loading.

Besides using the standard `sacplot.py` script, the user can modify its `getAxes` function in your own script to customize figure size and axes attributes. Script `egplot.py` (Figure 6) is such an example in which SAC p1, p2 styles and record section plotting are drawn in three axes in the same figure canvas. Run

```
egplot.py TA.[1-K]*Z -f1 -C
```

at command line to produce Figure 7. The ”-C” option uses random color for each seismogram. The ”-f1” option fills the positive signals of waveform with less transparency. In the script, ”opts.ynorm” sets the waveform normalization and ”opts.reltime=0” sets the time axis relative to time pick t0.

An improvement over SAC is that the program outputs the filename when the seismogram is clicked on by the mouse. This is enabled by the event handling API and is mostly introduced for use in SAC p2 style plotting when seismograms are plotted on top of each other. It is especially useful when a large number of seismograms create difficulties in labeling.

Another improvement is easier window zooming enabled by the SpanSelector widget and the event handling API. Select a time span by mouse clicking and dragging to zoom in a waveform section. Press the ’z’ key to zoom out to the previous time range.

3.3.2 SAC Phase Picking

SAC plotting (`pysmo.aimbat.plotphase`) does not involve change in data files but phase picking (`pysmo.aimbat.pickphase`) does. A GUI is built for user to interactively pick phase arrival times. Figure 8 is an example screen shot running

```
sacppk.py 20110915.19310408.bhz.pkl -w
```

in the data example directory `<example-event-dir>`.

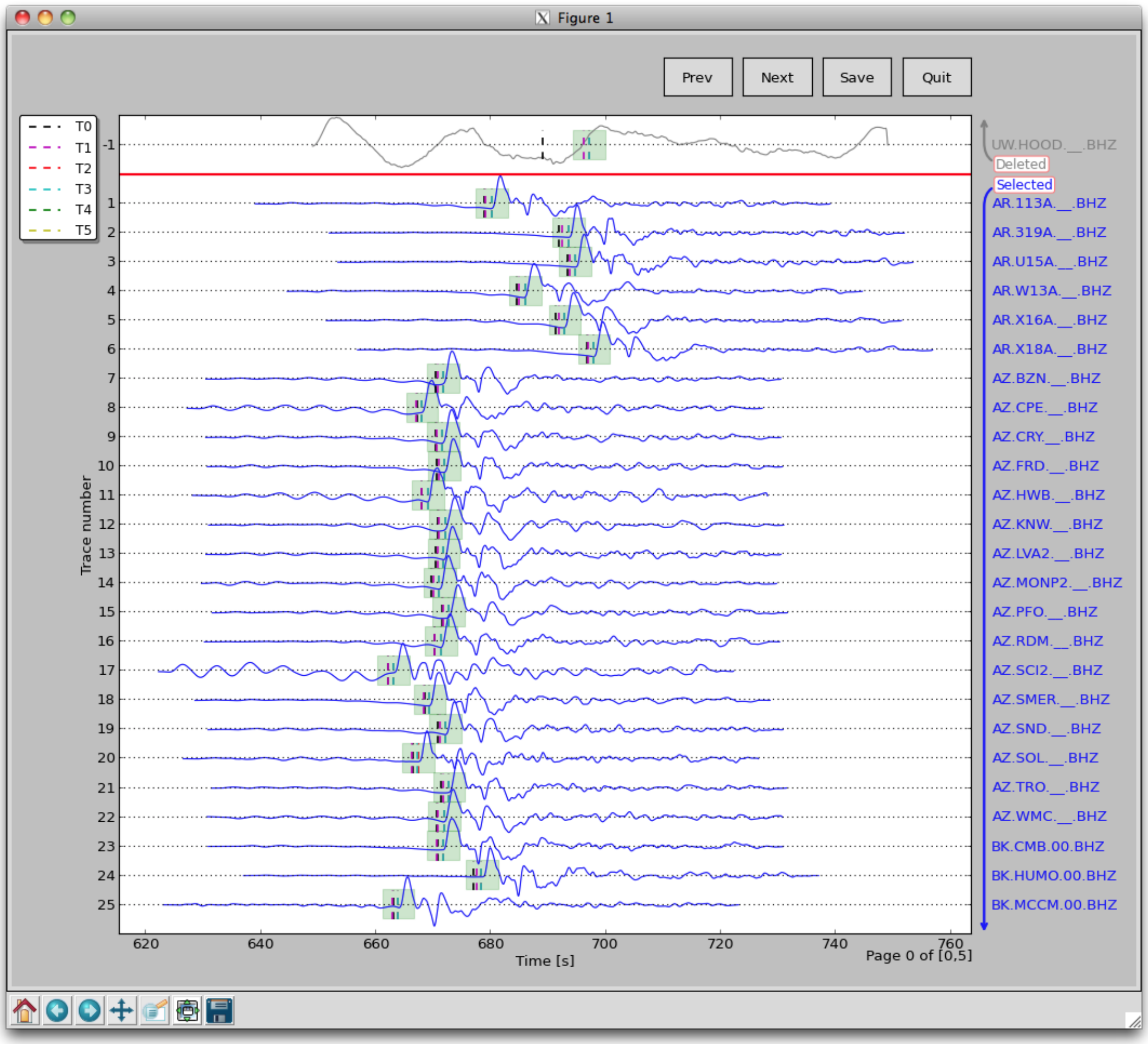


Figure 8: Screen shot of running "`sacppk.py 20110915.19310408.bhz.pkl -w`". First 25 out 162 selected seismograms and 1 deleted seismogram are plotted on the first page. Click the `Prev` and `Next` Buttons to navigate through the total 6 pages.

Following SAC convention, the user can set a time pick by pressing the 't' key and number keys '0-9'. The x location of the mouse position is saved to corresponding SAC headers 't0-t9'. Time window zooming in `pysmo.aimbat.pickphase` is implemented in the same way as in `pysmo.aimbat.plotphase` to replace SAC's combination of the 'x' key and mouse click. Zooming out key is set to 'z' because the 'o' key is used for another purpose by Matplotlib. The filename printing out by mouse clicking feature is also available in `pysmo.aimbat.pickphase`.

A major improvement over SAC is picking a time window in addition to time picks. Pressing the 'w' key to save the current time axis range to two user-defined SAC header variables. A transparent green span is plotted within the time window (Figure 8).

Another major improvement involves quality control with convenient operations to (de)select seismograms. In the GUI in Figure 8, there are two divisions of selected and deleted seismograms. Selected seismograms with a positive trace number are displayed with blue wiggles, while deleted seismograms with negative trace numbers are plotted in gray. The user can simply click on a certain seismogram to switch the selection status, either to exclude it or bring it back for inclusion. The trace selection status is stored in a user-defined SAC header variable.

In SAC, command "ppk p 10" plots 10 seismograms on each page. Pressing the 'b' and 'n' keys to navigate through pages. The number of seismograms plotted on each page is controlled by command line option "-m maxsel maxdel" for `sacppk.py`. The `Prev` and `Next` Buttons are for page navigation and the `Save` Button saves the change in time picks and time window to files. The default values for maxsel and maxdel are 25 and 5, which means a maximum of 30 seismograms on each page. In Figure 8, there are 26 seismograms on the first page because only 1 seismogram is deleted. On next page, there are 30 selected seismograms. To plot 50 seismograms on each page, run

```
sacppk.py 20110915.19310408.bhz.pkl -w -m 45 5
```

and there would be 4 total pages and 13 seismograms on the last page.

To plot seismograms relative to time pick t0 and fill the positive and negative wiggles of waveform, run

```
sacppk.py 20110915.19310408.bhz.pkl -w -r0 -f1
```

To sort seismograms by epicentral distance in increase and decrease orders, run

```
sacppk.py 20110915.19310408.bhz.pkl -w -sdist
sacppk.py 20110915.19310408.bhz.pkl -w -sdist-
```

Sorting by azimuth and back-azimuth is similar:

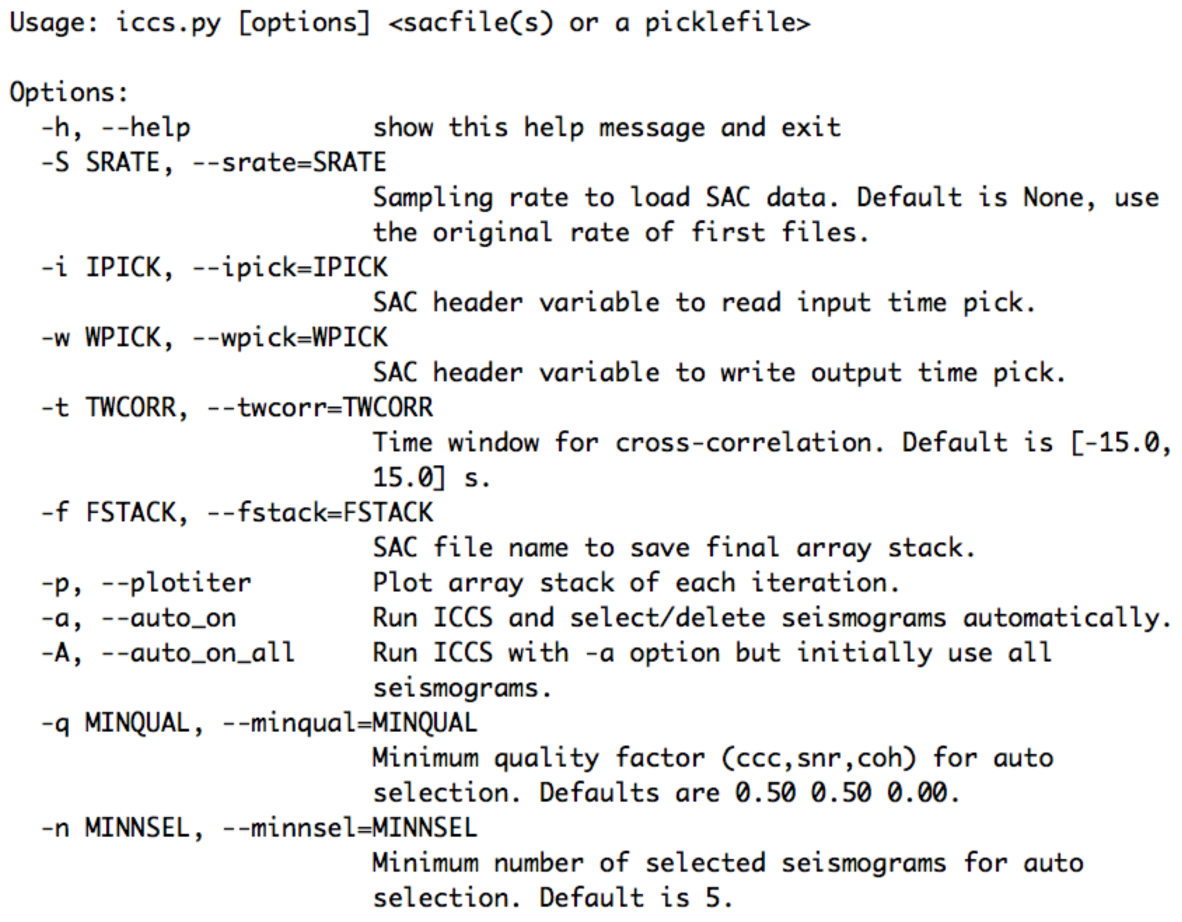
```
sacppk.py 20110915.19310408.bhz.pkl -w -saz
sacppk.py 20110915.19310408.bhz.pkl -w -sbaz
```

3.4 Measuring Teleseismic Body Wave Arrival Times

The core idea in using AIMBAT to measure teleseismic body wave arrival times has two parts: automated phase alignment and interactive quality control. The first part reduces user processing time and the second part retains valuable user inputs.

3.4.1 Automated Phase Alignment

The ICCS algorithm calculates an array stack from predicted time picks, cross-correlates each seismogram with the array stack to find the time lags at maximum cross-correlation, then use the new time picks to update the array stack in an iterative process. The MCCC algorithm cross-correlates each possible pair of seismograms and uses a least-squares method to calculate an optimized set of relative arrival times. Our method is to combine ICCS and



```
Usage: iccs.py [options] <sacfile(s) or a picklefile>

Options:
  -h, --help            show this help message and exit
  -S SRATE, --srate=SRATE
                        Sampling rate to load SAC data. Default is None, use
                        the original rate of first files.
  -i IPICK, --ipick=IPICK
                        SAC header variable to read input time pick.
  -w WPICK, --wpick=WPICK
                        SAC header variable to write output time pick.
  -t TWCORR, --twcorr=TWCORR
                        Time window for cross-correlation. Default is [-15.0,
                        15.0] s.
  -f FSTACK, --fstack=FSTACK
                        SAC file name to save final array stack.
  -p, --plotiter        Plot array stack of each iteration.
  -a, --auto_on         Run ICCS and select/delete seismograms automatically.
  -A, --auto_on_all     Run ICCS with -a option but initially use all
                        seismograms.
  -q MINQUAL, --minqual=MINQUAL
                        Minimum quality factor (ccc,snr,coh) for auto
                        selection. Defaults are 0.50 0.50 0.00.
  -n MINNSEL, --minnsel=MINNSEL
                        Minimum number of selected seismograms for auto
                        selection. Default is 5.
```

Figure 9: Help message of the `iccs.py` script.

MCCC in a four-step procedure using four anchoring time picks ${}_0T_i$, ${}_1T_i$, ${}_2T_i$, and ${}_3T_i$:

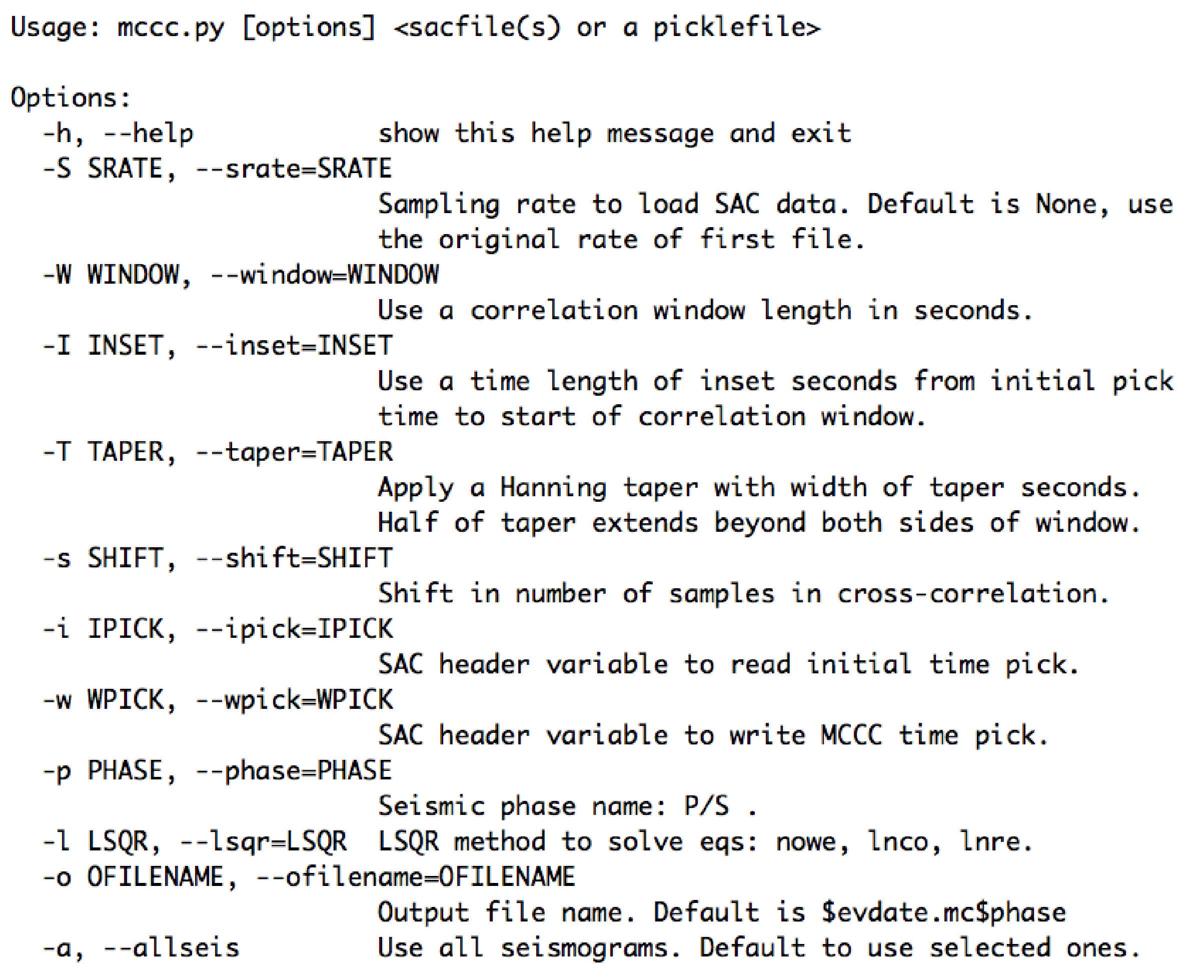
(a) Coarse alignment by ICCS

(b) Pick phase arrival at the array stack

(c) Refined alignment by ICCS

(d) Final alignment by MCCC

The input and output time picks for the steps (a), (c) and (d) and their corresponding SAC headers are listed in Table 2. The one-time manual phase picking at the array stack in step (b) allows the measurement of absolute arrival times. The detailed methodology and procedure can be found in *Lou et al.* (2012).



```
Usage: mccc.py [options] <sacfile(s) or a picklefile>

Options:
  -h, --help            show this help message and exit
  -S SRATE, --srate=SRATE
                        Sampling rate to load SAC data. Default is None, use
                        the original rate of first file.
  -W WINDOW, --window=WINDOW
                        Use a correlation window length in seconds.
  -I INSET, --inset=INSET
                        Use a time length of inset seconds from initial pick
                        time to start of correlation window.
  -T TAPER, --taper=TAPER
                        Apply a Hanning taper with width of taper seconds.
                        Half of taper extends beyond both sides of window.
  -s SHIFT, --shift=SHIFT
                        Shift in number of samples in cross-correlation.
  -i IPICK, --ipick=IPICK
                        SAC header variable to read initial time pick.
  -w WPICK, --wpick=WPICK
                        SAC header variable to write MCCC time pick.
  -p PHASE, --phase=PHASE
                        Seismic phase name: P/S .
  -l LSQR, --lsqr=LSQR  LSQR method to solve eqs: nowe, lnco, lnre.
  -o OFILENAME, --ofilename=OFILENAME
                        Output file name. Default is $evdate.mc$phase
  -a, --allseis         Use all seismograms. Default to use selected ones.
```

Figure 10: Help message of the `mccc.py` script.

Step	Algorithm	Input			Output	
		Time Window	Time Pick	Time Header	Time Pick	Time Header
(a)	ICCS	W_a	${}_0T_i$	**T0**	${}_1T_i$	**T1**
(c)	ICCS	W_b	${}_2T_i'$	**T2**	${}_2T_i$	**T2**
(d)	MCCC	W_b	${}_2T_i$	**T2**	${}_3T_i$	**T3**

Table 2: Time picks and their SAC headers used in the procedure for measuring teleseismic body wave arrival times.

The ICCS and MCCC algorithms are implemented in two modules `pysmo.aimbat.algiccs` and `pysmo.aimbat.algmccc`, and can be executed in scripts `iccs.py` and `mccc.py`, respectively. Usages are displayed in Figures 9 and 10.

3.4.2 Interactive Quality Control and Graphical User Interface

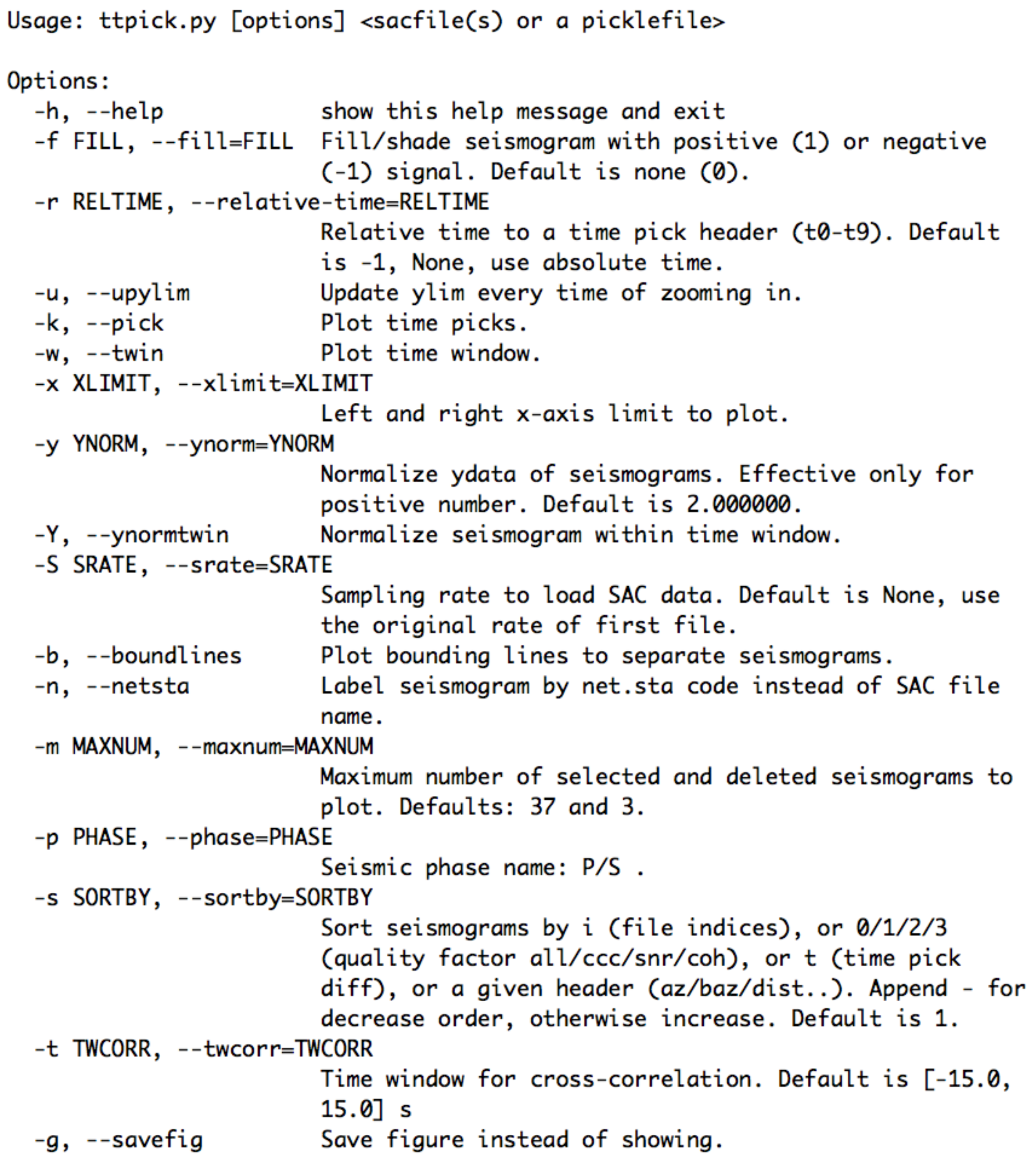
In practical data processing, there is a constant need of seismogram quality control, which is mixed with the procedure described above. ICCS steps (a), (b), and (c) are likely to be applied multiple times after removing low quality seismograms. To facilitate quality control, ICCS calculates three quality factors for each seismogram: cross-correlation coefficient (CCC), signal-to-noise ratio (SNR), and time domain coherence (COH). The ”-a” and ”-A” modes of `iccs.py` can remove seismograms with low qualities and rerun ICCS until all seismograms meet the minimum requirements specified by the ”-q” option.

A GUI is also designed to run both ICCS and MCCC and perform interactive quality control. It is implemented in module `pysmo.aimbat.qualctrl` and script `ttpick.py` based on `pysmo.aimbat.pickphase` and four Buttons for the four-step procedure. The quality control operation is the same as `sacppk.py`. The user can interactively switch the selection status of a seismogram by mouse clicking on the waveform. In order to efficiently delete low quality seismograms, different attributes listed below are used as sorting criteria:

- -s 0: sort by user-defined weighted-average of CCC, SNR and COH
- -s 1: sort by CCC
- -s 2: sort by SNR
- -s 3: sort by COH
- -s t: sort by time pick difference
- -s az: sort by azimuth

- -s baz: sort by back-azimuth
- -s dist: sort by epicentral distance

Other SAC headers also work with the ”-s” option. Default sorting order is increase. Append ”-” to make it decrease order, such as ”-s t-”. Sorting by time pick difference $({}_2T_i - {}_0T_i)$ in both increase and decrease orders are useful in detecting cycle-skipping.



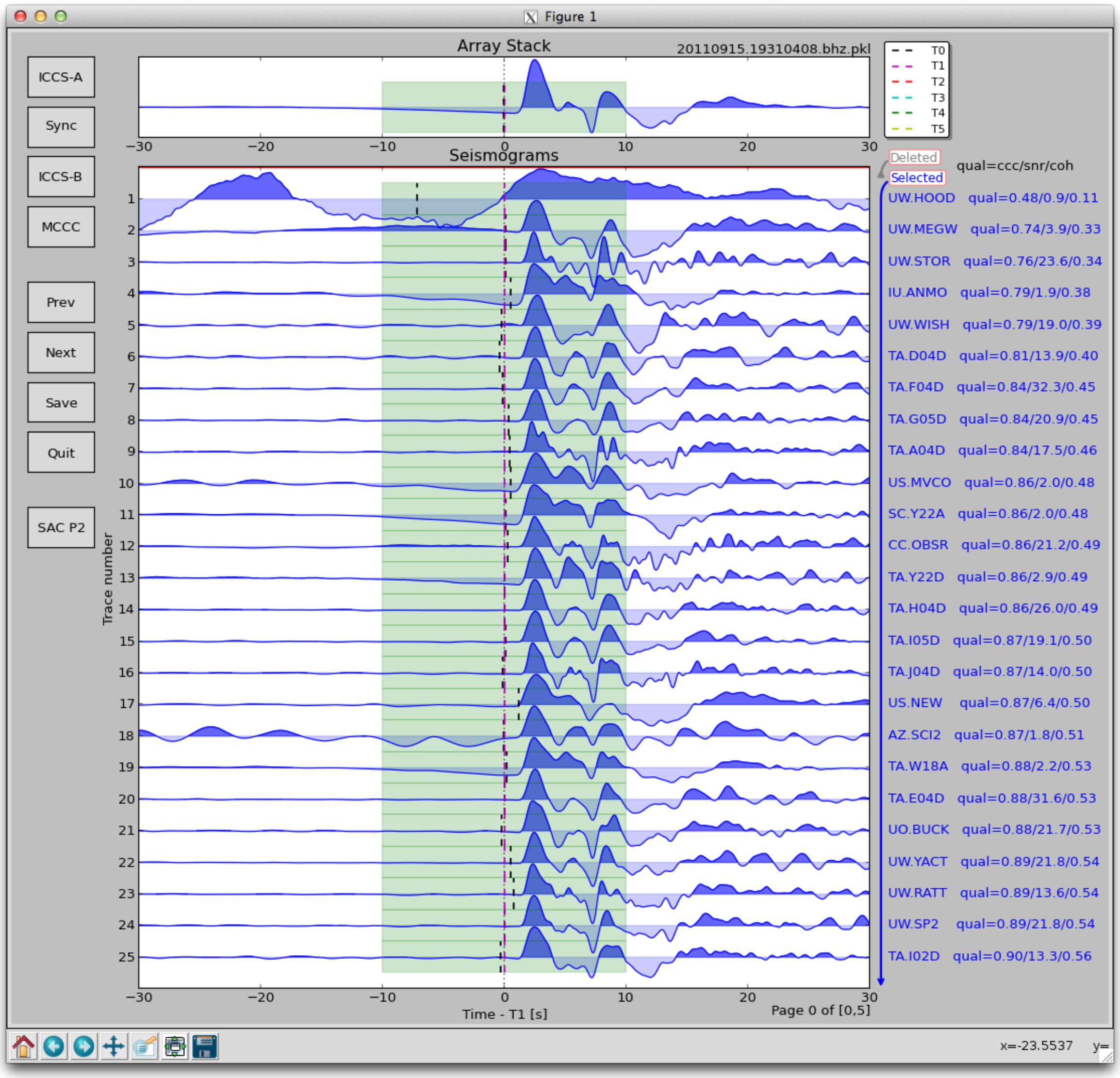
```
Usage: ttpick.py [options] <sacfile(s) or a picklefile>

Options:
  -h, --help            show this help message and exit
  -f FILL, --fill=FILL  Fill/shade seismogram with positive (1) or negative
                        (-1) signal. Default is none (0).
  -r RELTIME, --relative-time=RELTIME
                        Relative time to a time pick header (t0-t9). Default
                        is -1, None, use absolute time.
  -u, --upylim          Update ylim every time of zooming in.
  -k, --pick            Plot time picks.
  -w, --twin            Plot time window.
  -x XLIMIT, --xlimit=XLIMIT
                        Left and right x-axis limit to plot.
  -y YNORM, --ynorm=YNORM
                        Normalize ydata of seismograms. Effective only for
                        positive number. Default is 2.000000.
  -Y, --ynormtwin       Normalize seismogram within time window.
  -S SRATE, --srate=SRATE
                        Sampling rate to load SAC data. Default is None, use
                        the original rate of first file.
  -b, --boundlines      Plot bounding lines to separate seismograms.
  -n, --netsta          Label seismogram by net.sta code instead of SAC file
                        name.
  -m MAXNUM, --maxnum=MAXNUM
                        Maximum number of selected and deleted seismograms to
                        plot. Defaults: 37 and 3.
  -p PHASE, --phase=PHASE
                        Seismic phase name: P/S .
  -s SORTBY, --sortby=SORTBY
                        Sort seismograms by i (file indices), or 0/1/2/3
                        (quality factor all/ccc/snr/coh), or t (time pick
                        diff), or a given header (az/baz/dist..). Append - for
                        decrease order, otherwise increase. Default is 1.
  -t TWCORR, --twcorr=TWCORR
                        Time window for cross-correlation. Default is [-15.0,
                        15.0] s
  -g, --savefig         Save figure instead of showing.
```

Figure 11: Help message of the `ttpick.py` script.

To start the GUI, run

```
ttpick.py 20110915.19310408.bhz.pkl -t -10 10 -s1
```

in the data example directory `<example-event-dir>` and the screen shot is displayed in Figure 12. The "-t -10 10" option specifies an initial time window of (-10,10) seconds around the initial time pick to run ICCS. More screen shots along the data processing workflow are shown in Figures 13-16. See the captions for commands and descriptions.



```
qual=ccc/snr/coh
UW.HOOD   qual=0.48/0.9/0.11
UW.MEGW   qual=0.74/3.9/0.33
UW.STOR   qual=0.76/23.6/0.34
IU.ANMO   qual=0.79/1.9/0.38
UW.WISH   qual=0.79/19.0/0.39
TA.D04D   qual=0.81/13.9/0.40
TA.F04D   qual=0.84/32.3/0.45
TA.G05D   qual=0.84/20.9/0.45
TA.A04D   qual=0.84/17.5/0.46
US.MVCO   qual=0.86/2.0/0.48
SC.Y22A   qual=0.86/2.0/0.48
CC.OBSR   qual=0.86/21.2/0.49
TA.Y22D   qual=0.86/2.9/0.49
TA.H04D   qual=0.86/26.0/0.49
TA.I05D   qual=0.87/19.1/0.50
TA.J04D   qual=0.87/14.0/0.50
US.NEW    qual=0.87/6.4/0.50
AZ.SCI2   qual=0.87/1.8/0.51
TA.W18A   qual=0.88/2.2/0.53
TA.E04D   qual=0.88/31.6/0.53
UO.BUCK   qual=0.88/21.7/0.53
UW.YACT   qual=0.89/21.8/0.54
UW.RATT   qual=0.89/13.6/0.54
UW.SP2    qual=0.89/21.8/0.54
TA.I02D   qual=0.90/13.3/0.56
```

Figure 12: Screen shot of data processing initiated by running command `"ttpick.py 20110915.19310408.bhz.pkl -t -10 10 -s1 -x -30 30"`. Initial time window $W_a = (-10, 10)$ s. The "-x -30 30" option sets the time axis range to be (-30, 30) s. Seismograms are sorted by CCC.

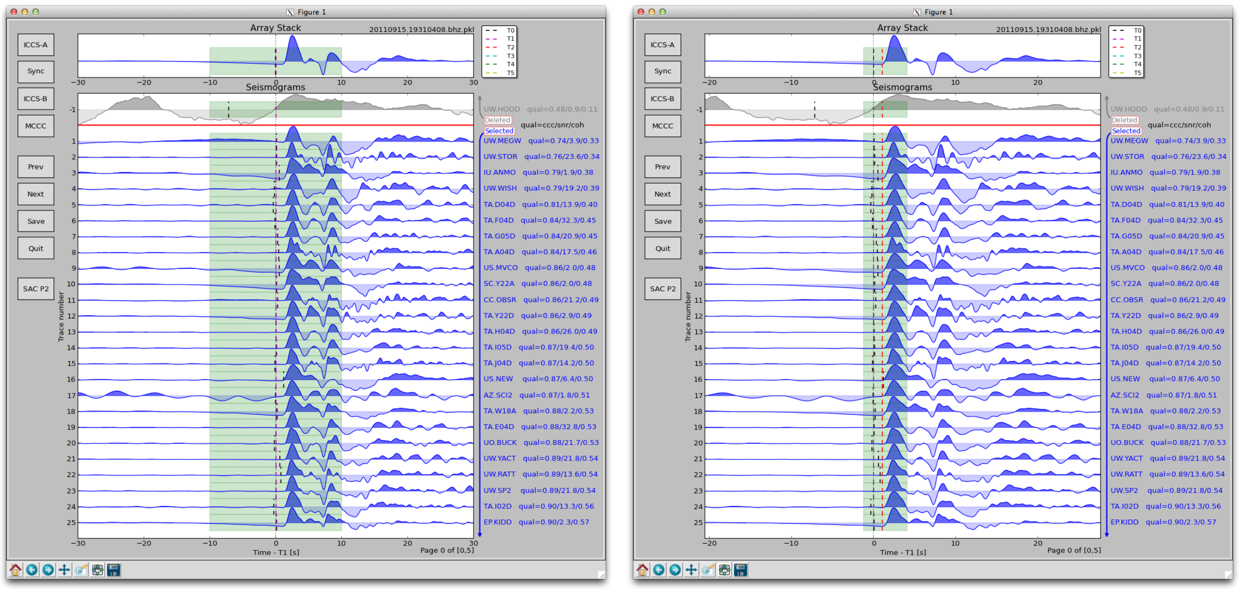


Figure 13: (A) Delete station UW.WOOD and rerun ICCS step (a) by clicking the `ICCS-A` Button. (B) Pick phase emergence on the array stack for measuring absolute arrival times. Choose a smaller time window $W_b = (-2.7, 2.9)$ s for refined alignment. Click the `Sync` Button to get ${}_2T_i'$ and time window for each seismogram.

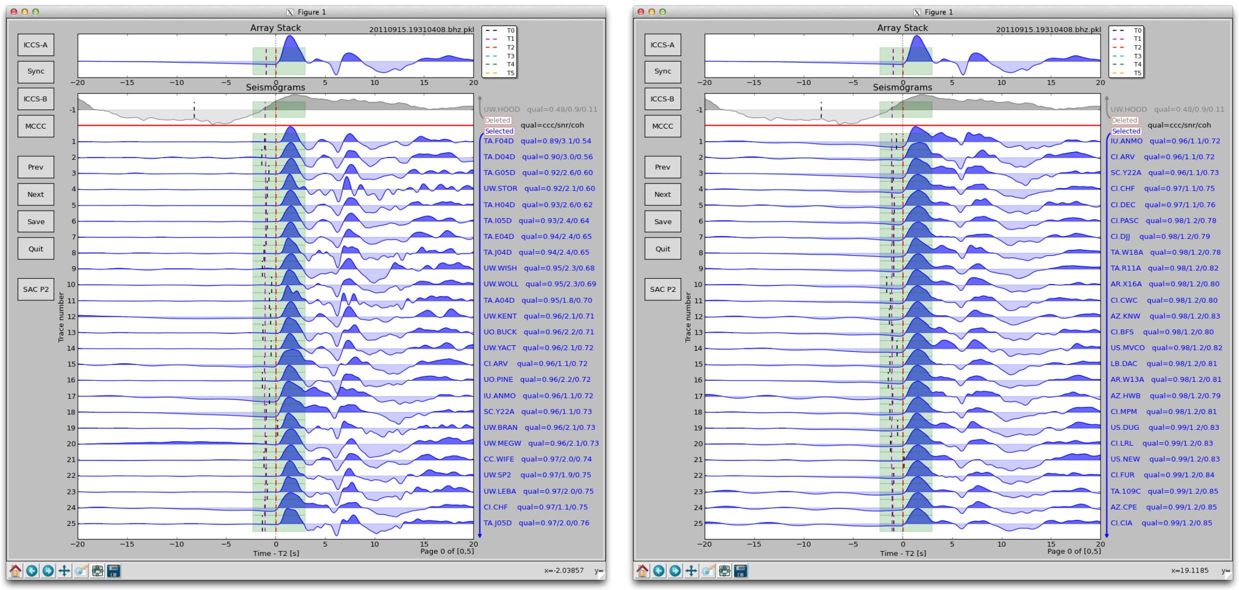


Figure 14: (A) Run ICCS step (c) by clicking the `ICCS-B` Button. Relative time pick is changed from T1 to T2. (B) Quit GUI and restart with `"ttpick.py 20110915.19310408.bhz.pkl -x -20 20 -r2 -s0"` which sorts seismograms by weighted-average of three quality factors.

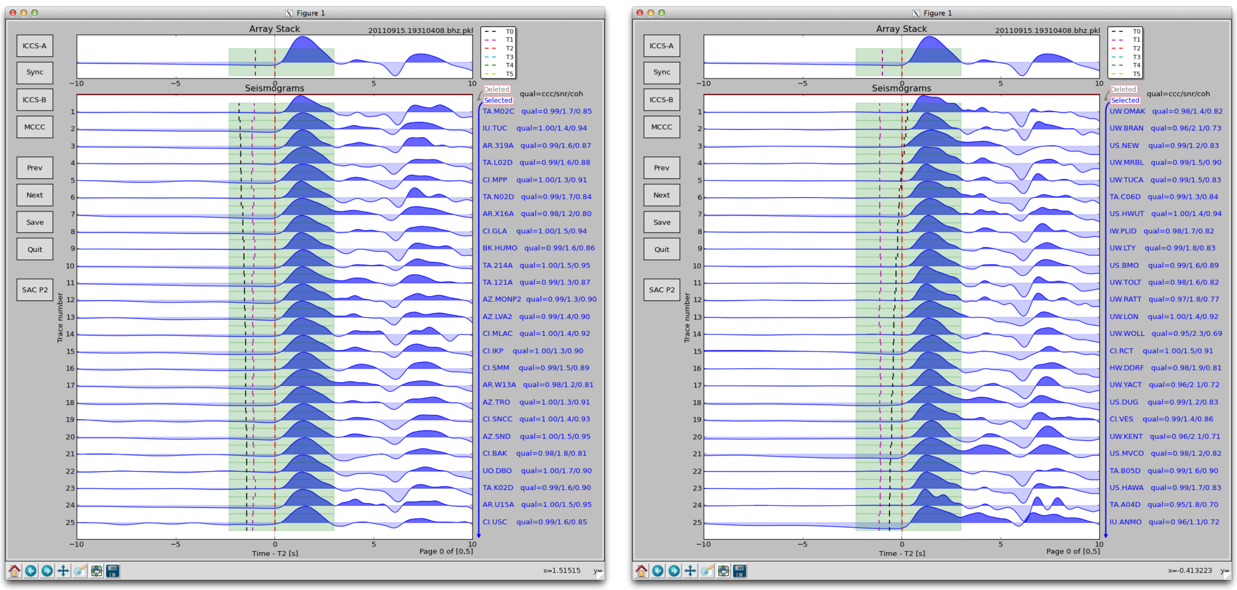


Figure 15: (A) Sort seismograms by time pick difference in increase order by running `"ttpick.py 20110915.19310408.bhz.pkl -x -10 10 -r2 -st"`. (B) Sort seismograms by time pick difference in decrease order by running `"ttpick.py 20110915.19310408.bhz.pkl -x -10 10 -r2 -st-"`

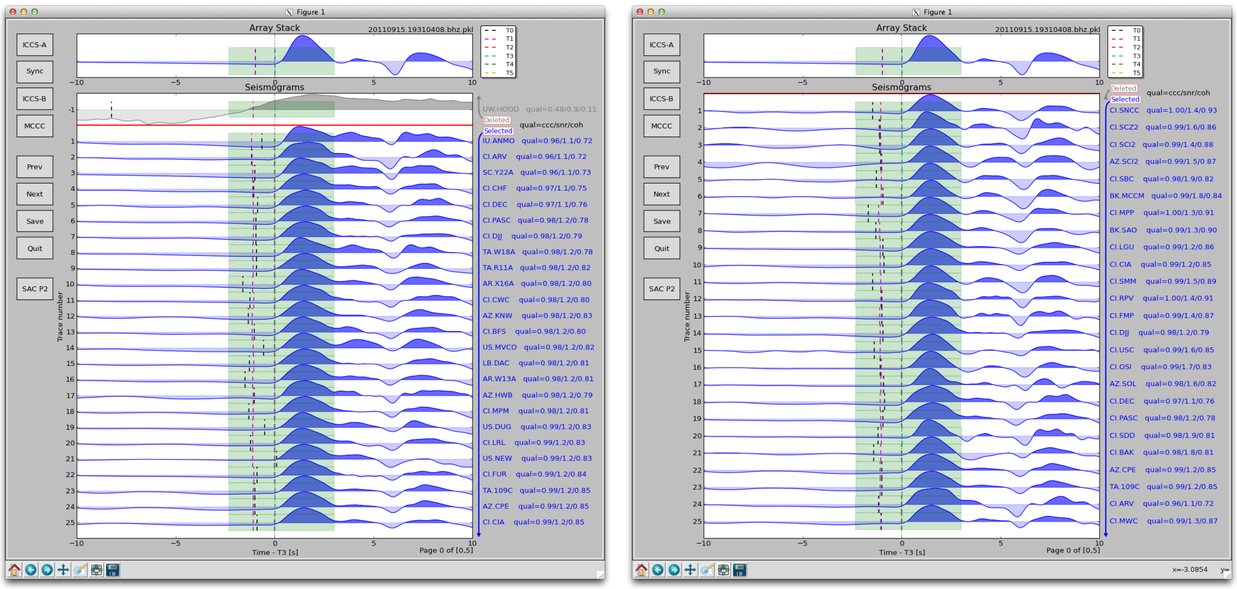


Figure 16: (A) Run MCCC by clicking the `MCCC` Button. Relative time pick is changed from T2 to T3. (B) Sort seismograms by time pick difference in decrease order by running `"ttpick.py 20110915.19310408.bhz.pkl -x -10 10 -r3 -sdist"`.

4 Appendix

A List of Files and Directories in the Packages

```
pysmo.sac:

    setup.py
    src/pysmo/sac
        sacio.py
        sacfunc.py
        sacmeth.py

pysmo.aimbat:

    setup.py
    src/pysmo/aimbat
        algiccs.py
        algmccc.py
        pickphase.py
        plotphase.py
        plotutils.py
        qualctrl.py
        qualsort.py
        sacpickle.py
        ttconfig.py
        ttdefaults.conf
        xcorr.py
        xcorrf.90
        xcorrf90.so
    scripts/
        egalign1.py
        egalign2.py
        egplot.py
        egsac.py
        iccs.py
        mccc.py
        sac2pkl.py
        sacp1.py
        sacp2.py
        sacpaz.py
        sacpbaz.py
        sacplot.py
        sacppk.py
        sacprs.py
        ttpick.py
    example/Event_2011.09.15.19.31.04.08/
```

References

Goldstein, P., D. Dodge, M. Firpo, and L. Minner (2003), SAC2000: Signal processing and analysis tools for seismologists and engineers, *International Geophysics*, *81*, 1613–1614.

Hunter, J. (2007), Matplotlib: A 2D Graphics Environment, *Computing in Science & Engineering*, *3*(9), 90–95.

Lou, X., S. van der Lee, and S. Lloyd (2012), A Python/Matplotlib Tool for Measuring Teleseismic Body Wave Arrival Times, *Seismological Research Letters, in revision.*

VanDecar, J. C., and R. S. Crosson (1990), Determination of teleseismic relative phase arrival times using multi-channel cross-correlation and least squares, *Bulletin of the Seismological Society of America*, *80*(1), 150–169.