# AIMBAT Documentation

***Release 1.0.8.dev15+g58e3189***

**Lay Kuan Loh, Xiaoting Lou, & Suzan van der Lee**

**Aug 22, 2023**

# CONTENTS:

CHAPTER

# ONE

# AIMBAT

## 1.1 Overview

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 [LouVanDerLeeLloyd2013]. It is based on a widely used method called MCCC (Multi-Channel Cross-Correlation) [VanDecarCrosson1990]. 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 [GoldsteinDodge2003] plotting and phase picking functionalities are replicated and enhanced.

Modules and scripts included in the AIMBAT package were developed using Python and its open-source modules on the Mac OS X platform since 2009. The original MCCC [VanDecarCrosson1990] code was transcribed into Python. The GUI of AIMBAT was inspired and initiated at the 2009 EarthScope USArray Data Processing and Analysis Short Course. AIMBAT runs on Mac OS X, Linux/Unix and Windows thanks to the platform-independent feature of Python. It has been tested on Mac OS 10.6.8 and 10.7, and Fedora 29.

For more information visit the project website or the Pysmo repository.

## 1.2 Documentation

For detailed installation and usage instructions see: https://aimbat.readthedocs.org.

## 1.3 Requirements

- Python version 3.6 or higher
- Fortran (optional, but highly recommended for better performance)

## 1.4 Installation

```
$ pip install pysmo.aimbat
```

## 1.5 Citation

AIMBAT: A Python/Matplotlib Tool for Measuring Teleseismic Arrival Times. Xiaoting Lou, Suzan van der Lee, and Simon Lloyd (2013), Seismol. Res. Lett., 84(1), 85-93, doi:10.1785/0220120033.

## 1.6 Authors' Contacts

- Xiaoting Lou Email: xlou at u.northwestern.edu
- Suzan van der Lee Email: suzan at northwestern.edu
- Simon Lloyd Email: simon at slloyd.net

## 1.7 Contributors

- Lay Kuan Loh

## 1.8 Licence

CHAPTER

TWO

# INSTALLATION AND UPGRADES

## 2.1 Requirements

### 2.1.1 Python

AIMBAT is built on top of standard Python and uses several extra Python libraries as well. We develop and test AIMBAT on Python versions 3.6 and newer. AIMBAT may run on older versions too, but we strongly suggest upgrading Python should you be running an older version (not just for AIMBAT!).

### 2.1.2 Compilers

#### 2.1.2.1 Fortran

Fortran is an optional dependency used to compile a fast cross-correlation routine during AIMBAT installation. AIMBAT will fall back to a Python cross-correlation routine if Fortran is not available, but it is slower. We therefore highly recommend installing a Fortran compiler (e.g. gfortran).

#### 2.1.2.2 C

While AIMBAT itself does not use C, some of the Python libraries it uses potentially require a C compiler during installation. This may vary depending on computer platform, or Python version/distribution used.

### 2.1.3 Operating System

AIMBAT is designed to run on UNIX like systems (e.g. Mac OSX and Linux). Installation on Windows is probably possible (since Python can be installed on almost any platform), though untested.

## 2.2 Installing AIMBAT

### 2.2.1 pip - Python package installer

AIMBAT is available as a package from the Python Package Index. This means it can be easily installed with the `pip` command (available by default since Python version 3.4).

**Caution:** It is possible to have multiple versions of Python installed on a computer. If this is the case, then there will also be multiple versions of the `pip` command. It is therefore important to use the `pip` command belonging to the Python version you intend to use for AIMBAT! Running `pip --version` will show you which Python version it belongs to.

**Note:** On some systems Python 2 and Python 3 are installed alongside eachother. Typically there is a `pip` command belonging to Python 2 and a `pip3` command belonging to Python 3.

### 2.2.2 conda users

If you are using conda to manage Python packages, we recommend installing AIMBAT dependencies with `conda` before installing AIMBAT with `pip`. To do so issue this command:

```
$ conda install scipy numpy matplotlib pyqtgraph pyyaml pyproj
```

**Note:** Similarly, if you are using a package manager on Linux, or something like brew or macports on OSX, you may want to install these dependencies (if available) via those mechanisms instead of `pip`.

### 2.2.3 Installing AIMBAT with pip

To install the latest stable version of AIMBAT and all dependencies not already installed, simply issue this command:

```
$ pip install pysmo.aimbat
```

**Caution:** Unless you know what you are doing, we recommend to *not* install AIMBAT with administrative priveliges (i.e. using sudo or the root account). If the above command fails due to insuffienct rights, run the same command with the `--user` flag:

```
$ pip install --user pysmo.aimbat
```

If you wish to install the latest developement version of AIMBAT *instead* of the stable release:

```
$ pip install git+https://github.com/pysmo/aimbat
```

**Note:** It is possible to install the stable release alongside the development version. Please read *Developing AIMBAT* for instructions.

### 2.2.4 Example Data

Get the repository data-example from Github. There is some example code inside *data-example/example_pkl_files* that will be needed for later demonstrations.

## 2.3 Upgrading AIMBAT

Upgrading AIMBAT with `pip` is done with the same command used to install, with the addition of the `-U` flag:

```
$ pip install -U pysmo.aimbat
```

**Note:** If you used the `--user` flag during installation you also need to use it while upgrading

## 2.4 Uninstalling AIMBAT

To remove aimbat from your system with `pip` run this command:

```
$ pip uninstall pysmo.aimbat
```

**Note:** Unfortunately `pip` currently does not remove dependencies that were automatically installed. We suggest running `pip list` to see the installed packages, which can then also be removed using `pip uninstall`

CHAPTER

THREE

# USING AIMBAT

## 3.1 Seismic Analysis Code (SAC)

AIMBAT uses Seismic Analysis Code (SAC) formatting for some of the files it runs and outputs. To get SAC, you will need to fill out a software request form available on the IRIS website.

## 3.2 SAC Input/Output procedures for AIMBAT

Aimbat converts SAC files to python pickle data structure to increase data processing efficiency by avoiding frequent SAC file I/O.

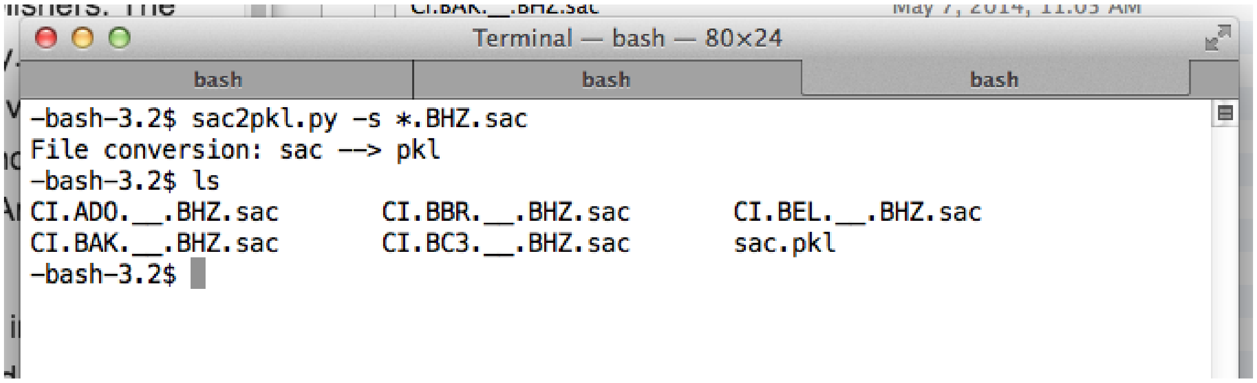
Reading and writing SAC files is done only once each before and after data processing, and intermediate processing is performed on python objects and pickles.

### 3.2.1 Converting from SAC to PKL files

Place the SAC files you want to convert to a pickle (PKL) file into the same folder. Suppose, for instance, they are BHZ channels. Note that the SAC files must be of the same channel. cd into that folder, and run:

```
aimbat-sac2pkl -s *.BHZ.sac
```

The output should be a PKL file in the same folder as the sac files.



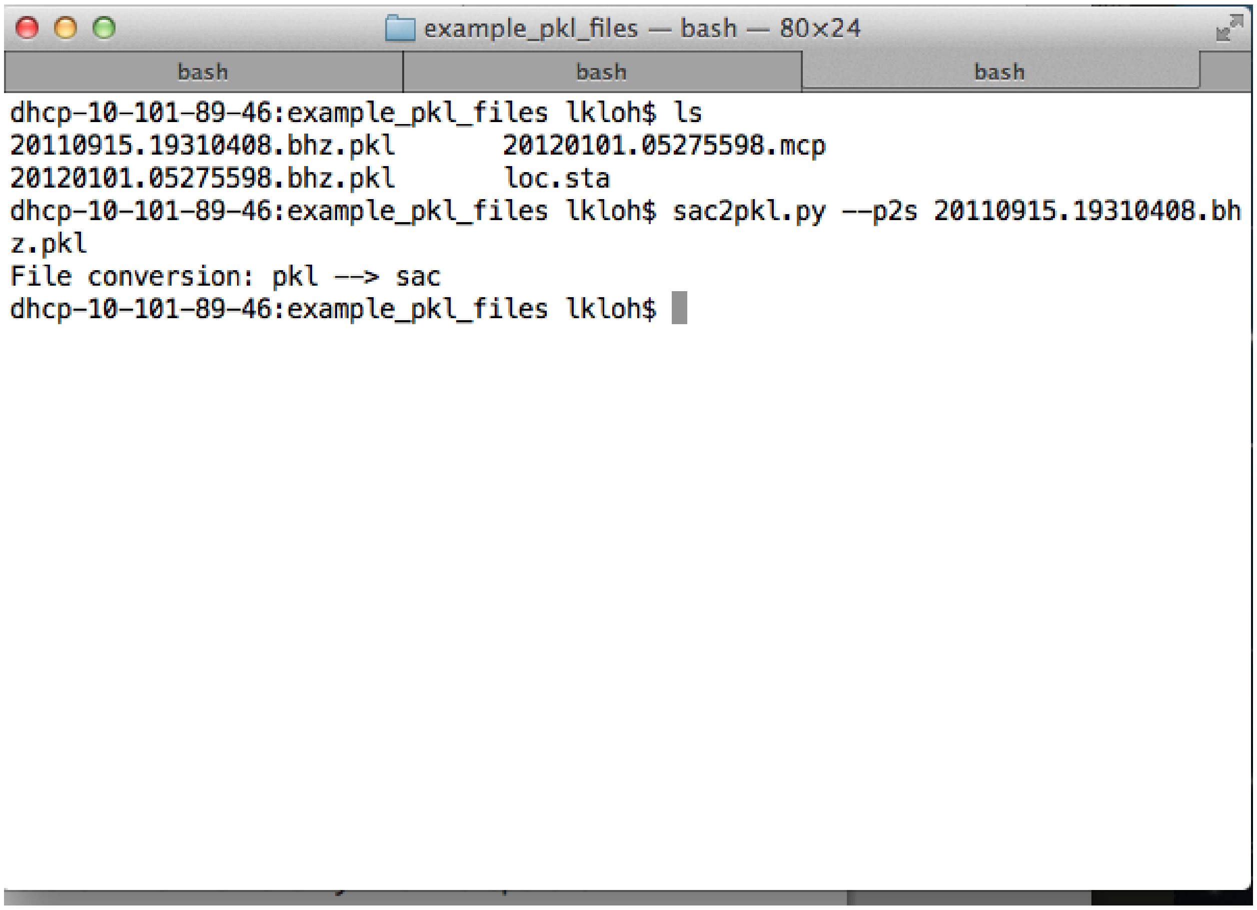
```
-bash-3.2$ sac2pkl.py -s *.BHZ.sac
File conversion: sac --> pkl
-bash-3.2$ ls
CI.ADO.__.BHZ.sac       CI.BBR.__.BHZ.sac       CI.BEL.__.BHZ.sac
CI.BAK.__.BHZ.sac       CI.BC3.__.BHZ.sac       sac.pkl
-bash-3.2$
```

### 3.2.2 Converting from PKL to SAC files

cd into the folder containing the PKL file that you wish to convert into SAC files, and run:

```
aimbat-sac2pkl --p2s <name-of-file>.pkl
```

The SAC files contained within will output into the same folder as the PKL file is stored in.



```
dhcp-10-101-89-46:example_pkl_files lkloh$ ls
20110915.19310408.bhz.pkl      20120101.05275598.mcp
20120101.05275598.bhz.pkl      loc.sta
dhcp-10-101-89-46:example_pkl_files lkloh$ sac2pkl.py --p2s 20110915.19310408.bh
z.pkl
File conversion: pkl --> sac
dhcp-10-101-89-46:example_pkl_files lkloh$ 
```

## 3.3 Parameter Configuration

### 3.3.1 Backend

Matplotlib works with six GUI (Graphical User Interface) toolkits:

1. WX
2. Tk
3. Qt(4)
4. FTK
5. Fltk
6. macosx

The GUI of AIMBAT uses the following to support interactive plotting:

1. GUI neutral widgets
2. GUI neutral event handling API (Application Programming Interface)

Visit these pages for an explanation of the backend and how to customize it.

AIMBAT uses the default toolkit `Tk` and backend `TkAgg`.

In the latest version, user does not need to setup the backend for the SAC plotting functions.

### 3.3.2 Configuration File

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

Python scripts in the `<pkg-install-dir>/pysmo-aimbat-0.1.2/scripts` can be executed from the command line. The command line arguments are parsed by the optparse module to improve the scripts' exitability. 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.

#### 3.3.2.1 Example of AIMBAT configuration file *ttdefaults.conf*

| 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] or Align/Refine | |
|---|---|
| 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: xcorr_full/fast/faster, reverse polarity allowed xcorr_full/fast/faster_polarity, reverse polarity not allowed |
| 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 = fs-tack.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. |
| xcorr_modu = xcorrf90 | Module for calculating cross-correlation: xcorr for Numpy or xcorrf90 for Fortran |
| xcorr_func = xcorr_fast | Function for calculating cross-correlation: xcorr_full/fast/faster, reverse polarity allowed xcorr_full/fast/faster_polarity, reverse polarity not allowed |
| 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 |
| fhdrBand = kuser1 | SAC Header to store filter type |
| fhdrApply = kuser1 | SAC Header to store applying filter or not |
| fhdrRevPass = user4 | SAC Header to store reverse pass of filter |
| fhdrLowFreq = user5 | SAC Header to store low frequency of band pass filter |
| fhdrHighFreq = user6 | SAC Header to store high frequency of band pass filter |
| fhdrOrder = user7 | SAC Header to store order of band pass filter |
| fvalApply = 0 | Value of applying filter or not |
| fvalBand = bandpass | Value of filter type |
| fvalRevPass = 0 | Value of reverse pass filter |
| fvalLowFreq = 0.05 | Value of low frequency of band pass filter |
| fvalHighFreq = 2 | Value of high frequency of band pass |
| fvalOrder = 2 | Value oforder of band pass filter |

## 3.4 SAC Data Access

NOTE: All `.sac` files must include origin time, hypocenter, as well as station coordinates and elevation in their headers.

### 3.4.1 Python Object for SAC File

The `pysmo.core.sac` package is developed to read and write individual SAC files. The Python class `SacIO` of module `pysmo.core.sac.sacio` opens a SAC file and returns an object including data and all SAC header variables as their attributes. Modifications of object attributes are saved to file. It is written purely in Python so that it also runs with Jython.

#### 3.4.1.1 *egsac.py*

The `<pkg-install-dir>/example-scripts/egsac.py` script 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 data example directory `<pkg-install-dir>/data-example/Event_2011.09.15.19.31.04.080`, hereafter referred to as *<example-event-dir>*.

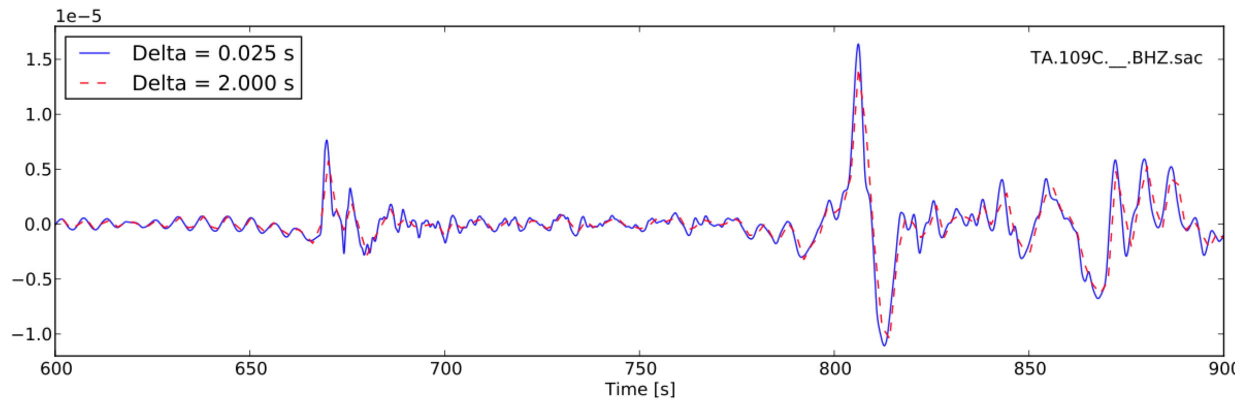
```python
from pysmo.core.sac import SacIO
from numpy import linspace, array
from scipy import signal
import matplotlib.pyplot as plt
import matplotlib.transforms as transforms

# read sac file:
ifilename = 'TA.109C.__.BHZ'
sacobj = SacIO.from_file(ifilename)
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.pdf', format='pdf')
plt.show()
```

### 3.4.1.2 Resampling Seismograms

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 signalprocessing module.

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

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



### 3.4.2 Python Pickle for SAC Files

The `pysmo.core.sac.sacio` module converts SAC files to `SacIO` 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 `SacIO` 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 to the `sacfile` object. Essentially, the `sacdh` object is a copy of the `sacfile` object in the memory, except that SAC headers 't0-t9', 'user0-user9', and '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 `aimbat-sac2pkl` does the conversions between SAC files and Python pickles.

Its usage message can be printed out by running at command line:

```
aimbat-sac2pkl -h
```

and the result is displayed in the figure below. For example, in the data example directory `<example-event-dir>`, run:

```
aimbat-sac2pkl -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:

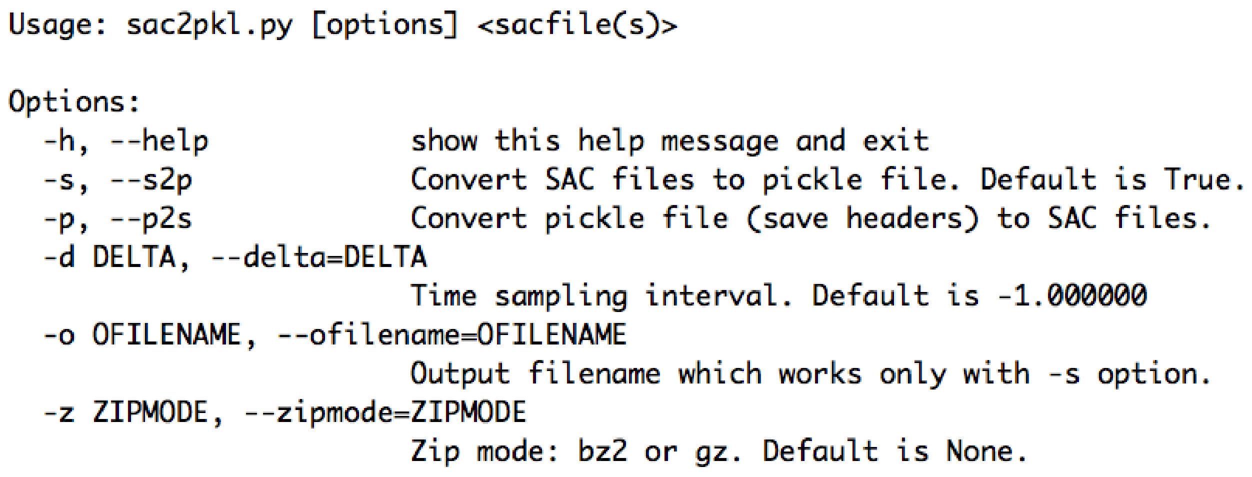
```
aimbat-sac2pkl -s *Z -o 20110915.19310408.bhz.pkl -d 0.025 -z gz
aimbat-sac2pkl -s *Z -o 20110915.19310408.bhz.pkl -d 0.025 -z bz2
```

at the cost of more CPU time.

After processing, run:

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

to convert the pickle file to SAC files.



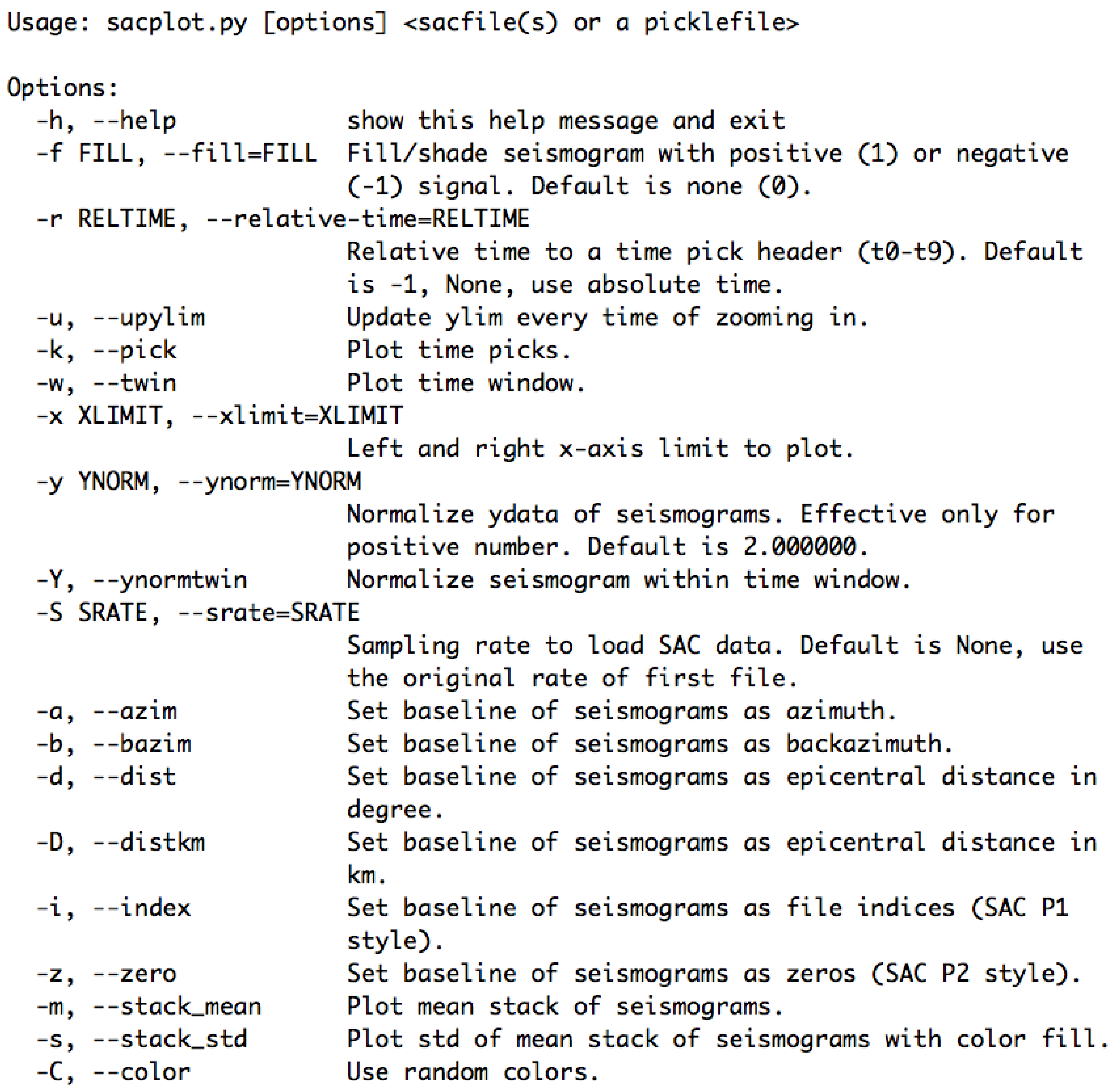
```
Usage: sac2pkl.py [options] <sacfile(s)>

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.
```

See the doc string of `pysmo.aimbat.sacpickle` by typing in a python console:

```python
from pysmo.aimbat import sacpickle
print(sacpickle.\_\_doc\_\_)
```

and also the documentation on pickle for more information about the Python data structure, pickling, and unpickling.

### 3.4.3 SAC Plotting and Phase Picking



```
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.
```

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.

They are implemented in two modules, `pysmo.aimbat.plotphase` and `pysmo.aimbat.pickphase`, which are used by corresponding scripts `aimbat-sacplot` and `aimbat-sacppk` executable at command line. Their help messages are displayed in the figures below.

```
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.
```

```
import matplotlib.pyplot as plt
import matplotlib.transforms as transforms
from pysmo.aimbat.plotphase import getDataOpts, sacp1, sacp2, sacprs

# figure axes
fig = plt.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.pdf', format='pdf')
plt.show()
```

### 3.4.4 SAC Plotting

Options "-i, -z, -d, -a, and -b" of `aimbat-sacplot` set the seismogram plotting baseline as file index, zero, epicentral distance in degrees, azimuth, and back-azimuth, respectively. The user can run `aimbat-sacplot` directly with the options, or run individual scripts `aimbat-sacp1`, `aimbat-sacp2`, `aimbat-sacprs`, `aimbat-sacpaz`, and `aimbat-sacpbaz`, 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:

```
aimbat-sacplot -i, aimbat-sacp1
aimbat-sacplot -z, aimbat-sacp2
aimbat-sacplot -d, aimbat-sacprs
```

(continues on next page)

(continued from previous page)

```
aimbat-sacplot -a, aimbat-sacpaz
aimbat-sacplot -b, aimbat-sacpbaz
```

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

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

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

```
aimbat-sacp1 TA.[1-K]*Z
```

or:

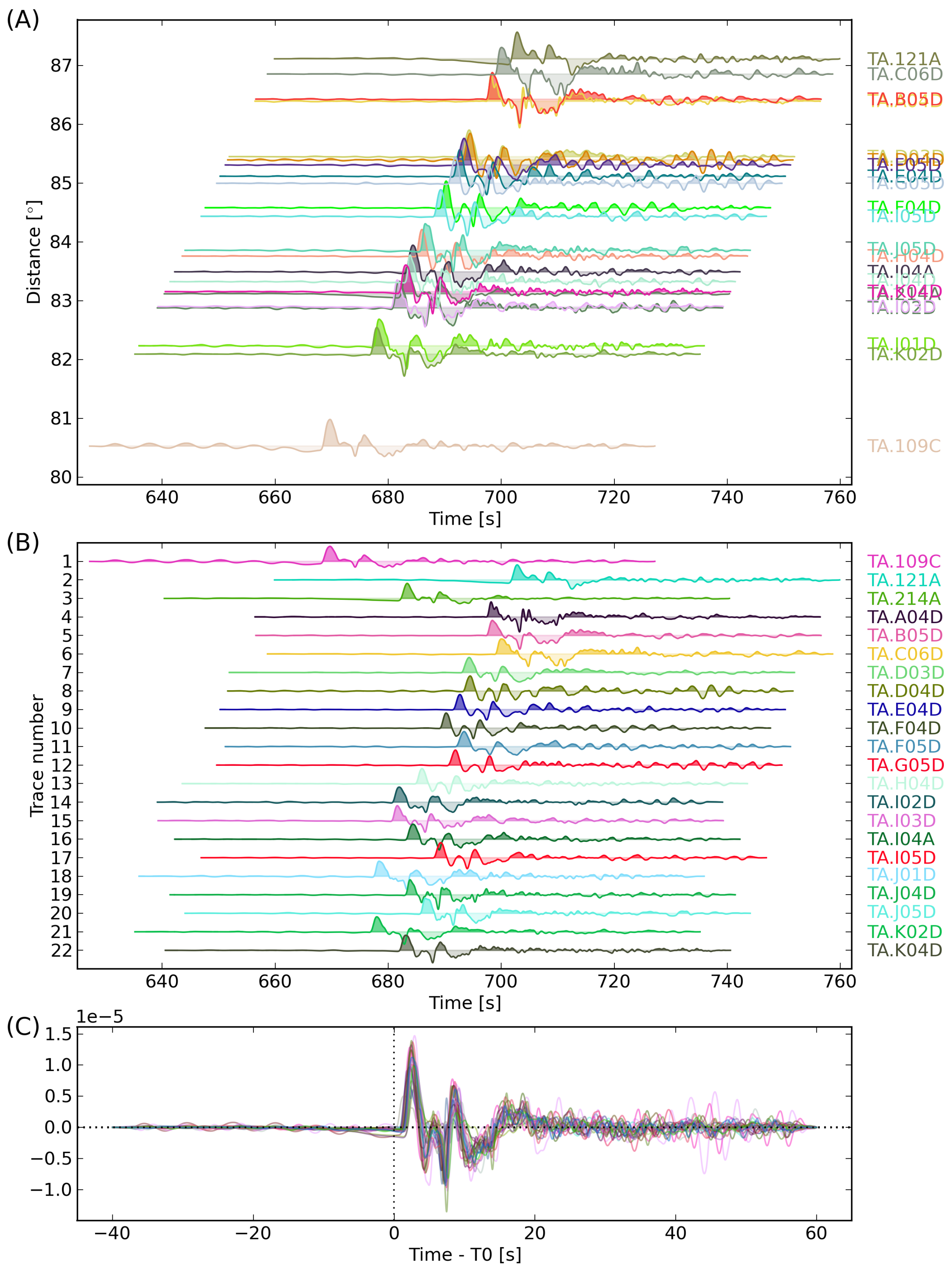
```
aimbat-sacp1 bhz.pkl
```

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

Besides using the standard `aimbat-sacplot` script, the user can modify its `getAxes` function in their own script to customize figure size and axes attributes. Script `egplot.py` 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 the figure below.



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.

## 3.5 SAC Phase Picking

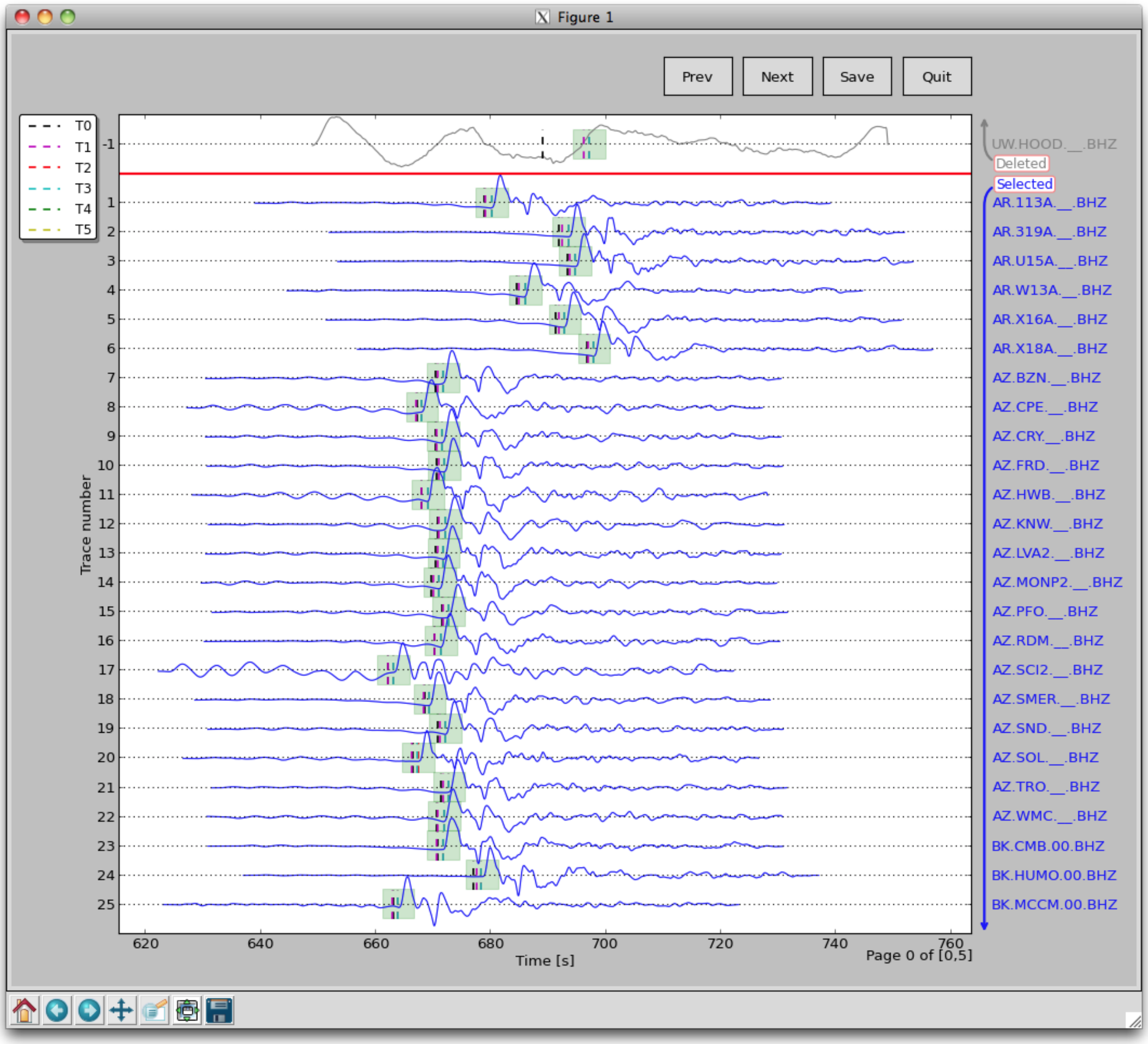
SAC plotting (`pysmo.aimbat.plotphase`) does not involve change in data files, but phase picking (`pysmo.aimbat.pickphase`) does. A GUI is built for the user to interactively pick phase arrival times. The figure below is an example screen shot running:

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

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

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, as shown in the figure below.



Another major improvement involves quality control with convenient operations to (de)select seismograms. In the GUI above, 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 `aimbat-sacppk`. 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 the figure displayed, there are 26 seismograms on the first page because only 1 seismogram is deleted. On the next page, there are 30 selected seismograms. To plot 50 seismograms on each page, run:

```
aimbat-sacppk 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:

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

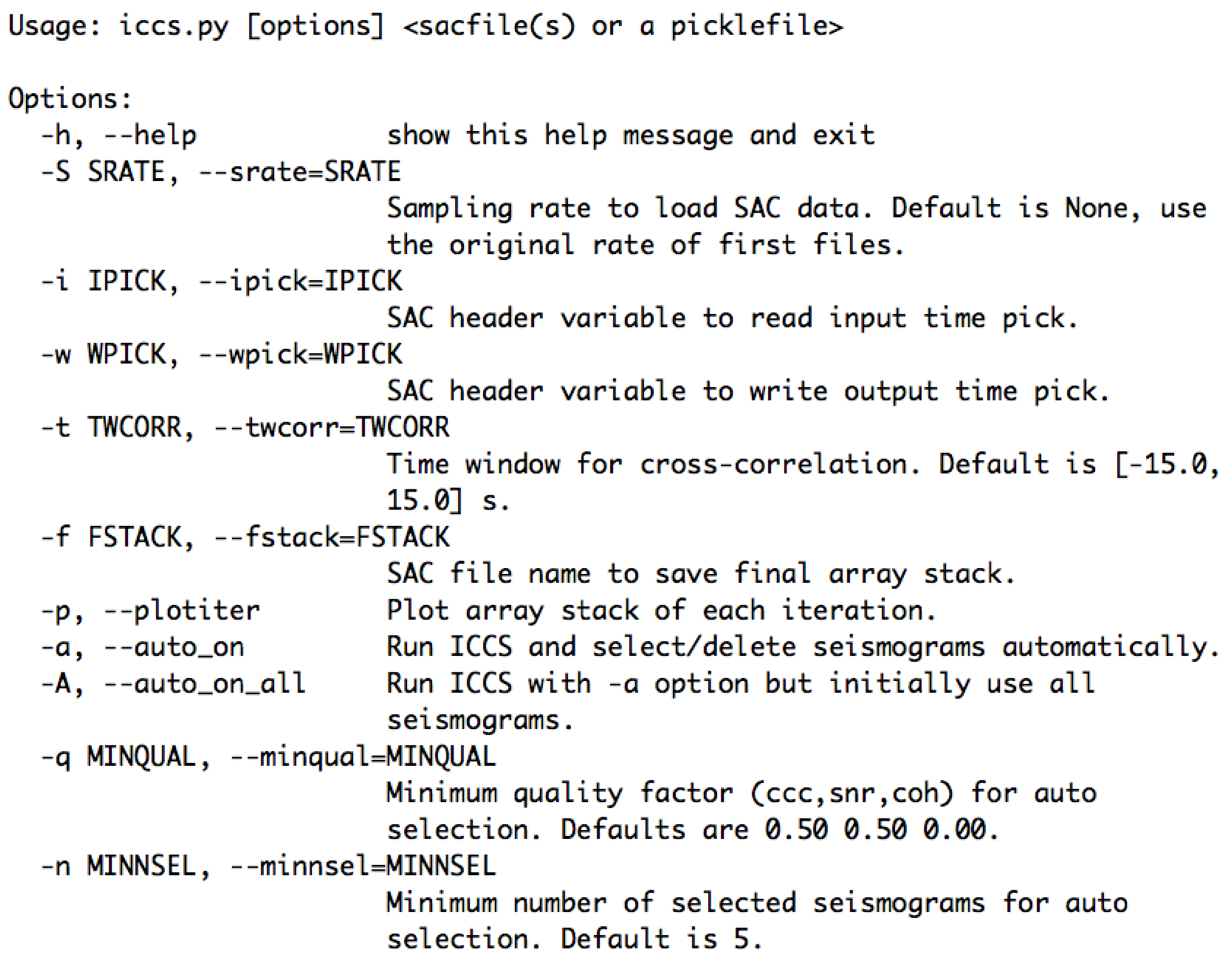
To sort seismograms by epicentral distance in increase and decrease orders, run:

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

Sorting by azimuth and back-azimuth is similar:

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

The help message of the `aimbat-iccs` script is shown below:



```
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.
```

The help message of the `aimbat-mccs` script is shown below:

```
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.
```

## 3.6 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, to reduce user processing time, and
- interactive quality control, to retain valuable user inputs.

### 3.6.1 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 uses 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 combines ICCS and 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 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 [LouVanDerLee2013].

Table 1: Time picks and their SAC headers used in the procedure for measuring teleseismic body wave arrival times.

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

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.

## 3.6.2 Picking Travel Times

This section explains how to run the program `aimbat-ttpick` to get the travel times you want.

### 3.6.2.1 Getting into the right directory

In the terminal, cd into the directory with all of the `pkl` files you want to run. You want to run either the `BHT` or `BHZ` files. `BHT` files are for S-waves and `BHZ` files are for P-waves. `PKL` is a bundle of `SAC` files. Each `SAC` file is a seismogram, but since there may be many seismograms from various stations for each event, we bundle them into a `PKL` file so we only have to import one file into AIMBAT, not a few hundred of them.

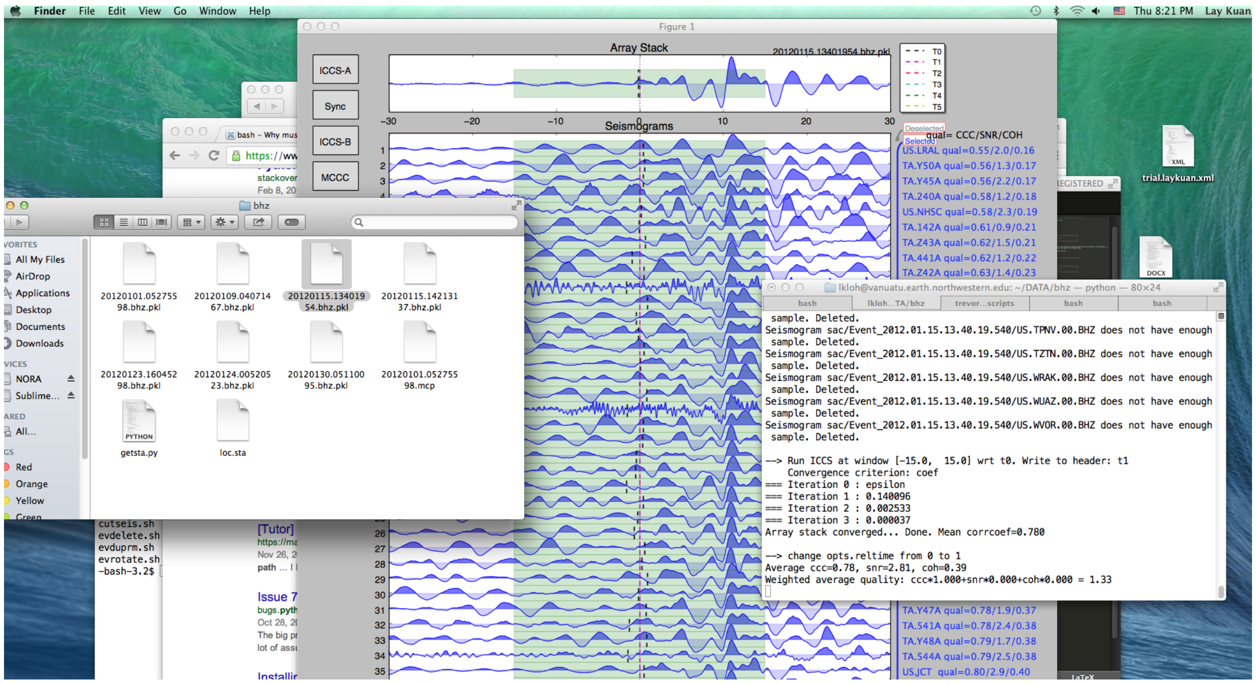
### 3.6.2.2 Running aimbat-ttpick

Run `aimbat-ttpick -p P <path-to-pkl-file>` for `BHZ` files or `aimbat-ttpick -p S <path-to-pkl-file>` for `BHT` files. A GUI should pop up if you successfully ran it. Note that if you click on the buttons, they will not work until you move your mouse off them; this is a problem we are hoping to fix.

You can get some example data to test this out by downloading the Github repository data-example. Now, cd into the folder *example_pkl_files*, which has several pickle files for seismic events. Type:

```
aimbat-ttpick -p P 20110915.19310408.bhz.pkl
```

and a python GUI should pop up.



```
US.LRAL qual=0.55/2.0/0.16
TA.Y50A qual=0.56/1.3/0.17
TA.Y45A qual=0.56/2.2/0.17
TA.240A qual=0.58/1.2/0.18
US.NHSC qual=0.58/2.3/0.19
TA.142A qual=0.61/0.9/0.21
TA.Z43A qual=0.62/1.5/0.21
TA.441A qual=0.62/1.2/0.22
TA.Z42A qual=0.63/1.4/0.23
```

```
 sample. Deleted.
Seismogram sac/Event_2012.01.15.13.40.19.540/US.TPNV.00.BHZ does not have enough
 sample. Deleted.
Seismogram sac/Event_2012.01.15.13.40.19.540/US.TZTN.00.BHZ does not have enough
 sample. Deleted.
Seismogram sac/Event_2012.01.15.13.40.19.540/US.WRAK.00.BHZ does not have enough
 sample. Deleted.
Seismogram sac/Event_2012.01.15.13.40.19.540/US.WUAZ.00.BHZ does not have enough
 sample. Deleted.
Seismogram sac/Event_2012.01.15.13.40.19.540/US.WVOR.00.BHZ does not have enough
 sample. Deleted.

--> Run ICCS at window [-15.0,  15.0] wrt t0. Write to header: t1
    Convergence criterion: coef
=== Iteration 0 : epsilon
=== Iteration 1 : 0.140096
=== Iteration 2 : 0.002533
=== Iteration 3 : 0.000037
Array stack converged... Done. Mean corrcoef=0.780

--> change opts.reltime from 0 to 1
Average ccc=0.78, snr=2.81, coh=0.39
Weighted average quality: ccc*1.000+snr*0.000+coh*0.000 = 1.33
```

```
TA.Y47A qual=0.78/1.9/0.37
TA.541A qual=0.78/2.4/0.38
TA.Y48A qual=0.79/1.7/0.38
TA.544A qual=0.79/2.5/0.38
US.JCT  qual=0.80/2.9/0.40
```

At the top of the GUI is the scaled sum of all of the seismograms known as the array stack, which gives a characteristic waveform of the event for the stations involved. Beneath this is a page of seismograms, with the corresponding station and various quality factors listed on the right. `CCC` is the cross-correlation coefficient between that seismogram and the array stack, `SNR` is the signal-to-noise ratio, and `COH` is the coherence between that seismogram and the array stack.

#### 3.6.2.3 Initial deselection of bad seismograms

Bad seismograms are those whose waveforms look nothing like the array stack above. By default, the seismograms are sorted by quality, so bad seismograms will likely be at the top. In order to deselect these, click on the waveforms themselves (not the fill) and wait a second or two for them to turn gray. The user can develop criteria for which seismograms to deselect and which to keep. Simply deselecting all seismograms below a certain quality threshold can decrease time but may lead to good seismograms being deselected or bad seismograms remaining.

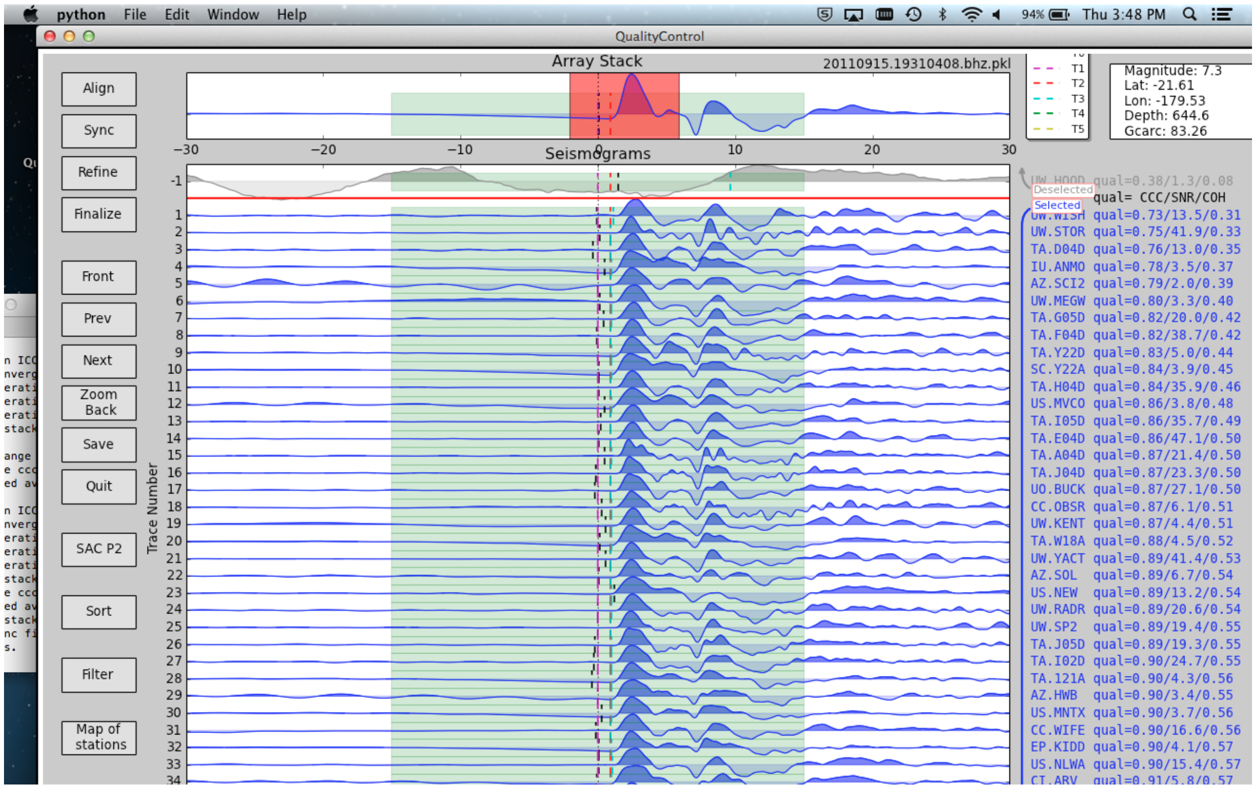
Remember to save your work periodically once you start picking your travel times. Otherwise, if AIMBAT crashes, you will lose your work.

#### 3.6.2.4 Align

The first step after deselecting seismograms is to press `Align`. This will recalculate both the array stack and the T1 picks for the seismograms, not including the deselected seismograms. Do not press `Align` after pressing `Sync` unless you wish to remove any `T2` picks that have been made.

#### 3.6.2.5 Sync, refine, and setting time window

After hitting the `Align` button, place the cursor on the array stack where the first motion of the seismogram, either up or down, occurs. Press `t` and `2` simultaneously on the keyboard to select the arrival time. Now press `Sync`. Use the mouse to drag and select the desired time window on the seismogram on the array stack. This time window is the portion of the seismogram on which cross-correlation will be run. The time window should begin 2-10 seconds before the first arrival and include a few seconds of the first motion of the waveform. The final time window should be smaller than the default window to increase the accuracy of the cross-correlation.



```
Magnitude: 7.3
Lat: -21.61
Lon: -179.53
Depth: 644.6
Gcarc: 83.26
```

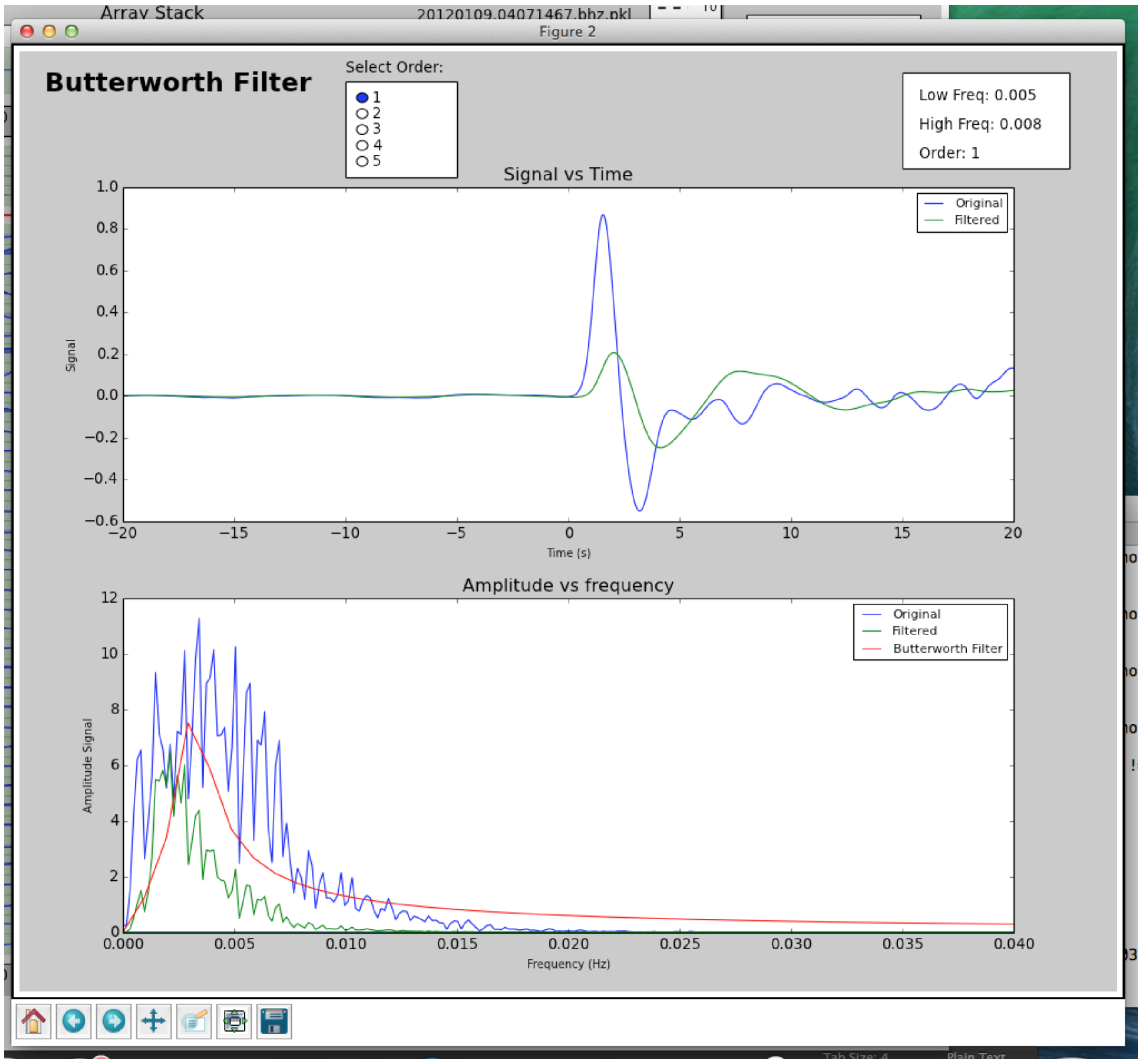
```
UW.HOOD qual=0.38/1.3/0.08
        qual= CCC/SNR/COH
UW.WISH qual=0.73/13.5/0.31
UW.STOR qual=0.75/41.9/0.33
TA.D04D qual=0.76/13.0/0.35
IU.ANMO qual=0.78/3.5/0.37
AZ.SCI2 qual=0.79/2.0/0.39
UW.MEGW qual=0.80/3.3/0.40
TA.G05D qual=0.82/20.0/0.42
TA.F04D qual=0.82/38.7/0.42
TA.Y22D qual=0.83/5.0/0.44
SC.Y22A qual=0.84/3.9/0.45
TA.H04D qual=0.84/35.9/0.46
US.MVCO qual=0.86/3.8/0.48
TA.I05D qual=0.86/35.7/0.49
TA.E04D qual=0.86/47.1/0.50
TA.A04D qual=0.87/21.4/0.50
TA.J04D qual=0.87/23.3/0.50
UO.BUCK qual=0.87/27.1/0.50
CC.OBSR qual=0.87/6.1/0.51
UW.KENT qual=0.87/4.4/0.51
TA.W18A qual=0.88/4.5/0.52
UW.YACT qual=0.89/41.4/0.53
AZ.SOL  qual=0.89/6.7/0.54
US.NEW  qual=0.89/13.2/0.54
UW.RADR qual=0.89/20.6/0.54
UW.SP2  qual=0.89/19.4/0.55
TA.J05D qual=0.89/19.3/0.55
TA.I02D qual=0.90/24.7/0.55
TA.121A qual=0.90/4.3/0.56
AZ.HWB  qual=0.90/3.4/0.55
US.MNTX qual=0.90/3.7/0.56
CC.WIFE qual=0.90/16.6/0.56
EP.KIDD qual=0.90/4.1/0.57
US.NLWA qual=0.90/15.4/0.57
CI.ARV  qual=0.91/5.8/0.57
```

Next, set the cursor over the array stack and press the `w` key. If the new time window has been saved, a message noting the new size of the time window will be printed in the terminal. The entire width of the x-axis is now colored green and will be stored as the time window to use for the cross-correlations. Press `Save headers only` if you wish to keep this time window for future applications.

Now press `Refine` and all the seismograms will align with the smaller time window. Note that the `Weighted average quality` printed to the terminal may decrease significantly, but this is likely due to the fact that the time window is smaller than the original.

#### 3.6.2.6 Filtering

If you wish to apply a filter to your data, hit the `Filter` button, and a window will pop up for you to use the Butterworth filter to filter your data.



Low Freq: 0.005
High Freq: 0.008
Order: 1

The defaults used for filtering are:

| Variable | Default |
|---|---|
| Order | 2 |
| Filter Type | Bandpass |
| Low Frequency | 0.05 Hz |
| High Frequency | 0.25 Hz |

You can change the order and filter type by selecting the option you want. In order to set corner frequencies for the filter, select the low frequency and the high frequency you want on the lower figure by clicking. Press `Apply` to filter the seismograms when you are satisfied with the filter parameters chosen.

#### 3.6.2.7 Finalize

Hit `Finalize` to run the multi-channel cross-correlation. Do not hit `Align` or `Refine` again, or all your previous picks will be written over. A warning will pop up to check if you really do want to hit these two buttons if you do click on them.

#### 3.6.2.8 SACP2 to check for outlier seismograms

Hit `SACP2` and go to the last figure, (d). Zoom in to have a better look. Zooming in doesn't always work well; close and reopen the `SACP2` window if there are problems.

Click on the outliers that stray from the main group of stacked seismograms. The terminal will output the names of the seismograms that you clicked on, so you can return to the main GUI window and readjust the travel times. Note: hitting `SACP2` before hitting `Finalize` will often cause AIMBAT to close, so make sure you have finalized before using SACP2.

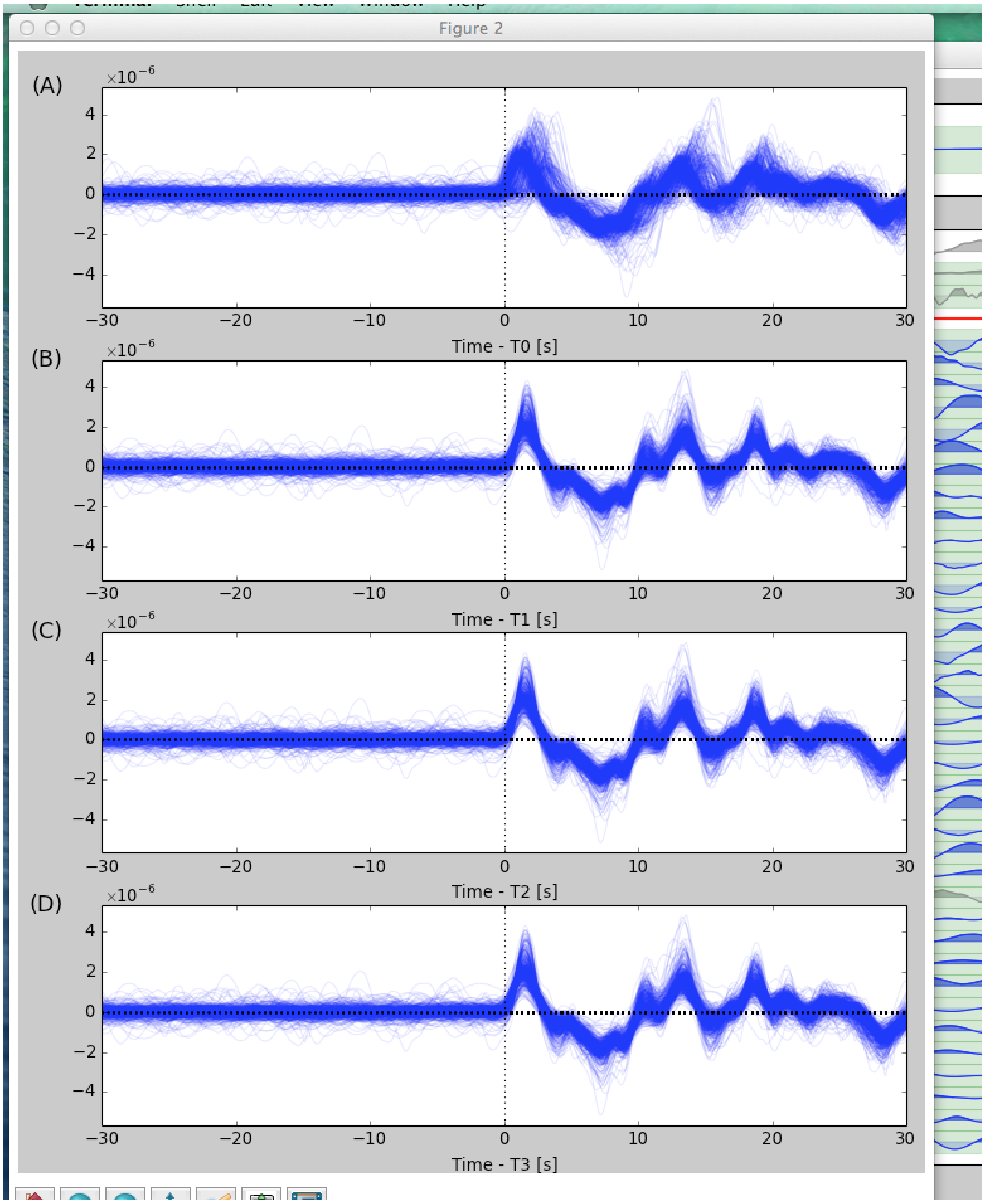


Figure 2
(A)
×10⁻⁶
Time - T0 [s]
(B)
Time - T1 [s]
(C)
Time - T2 [s]
(D)
Time - T3 [s]

#### 3.6.2.9 Go through the badly aligned seismograms and realign the travel times manually

By default, the worst seismograms are on the first page, and as you click through the pages, the quality of the seismograms gradually gets better. Keep using `t2` to realign the arrival times so that the peaks of all the seismograms are nicely aligned. Remember to zoom in to have a better look.

However, you may wish to sort the seismograms in alphabetical order or by azimuth so that you can find the bad seismogrrams and correct them more easily. Hit the `Sort` button and a window will pop up for you to choose which sorting method to use. In this case, choose `File` to sort the files by station name alphabetically, or choose `AZ` to sort the files by azimuth from the event epicenter. The seismograms are stretched to fit together, but they may be scaled differently.

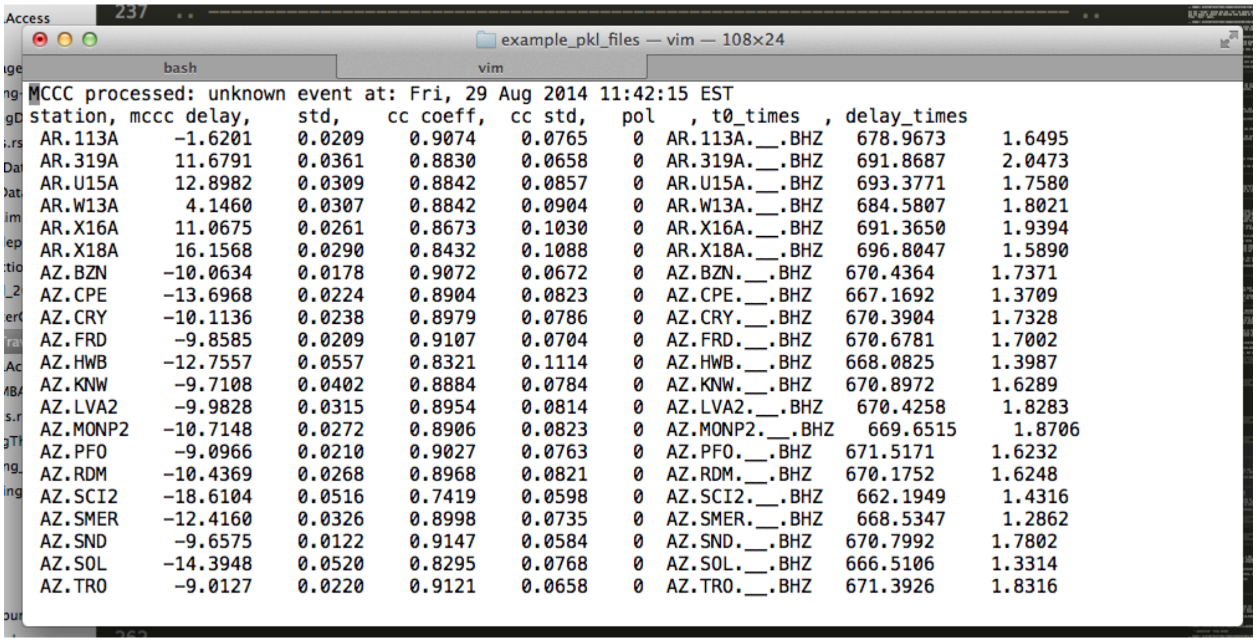
## 3.6.3 What the Alignments Stand For

- T0: Theoretical Arrival
- T1: Pick from initial cross correlation
- T2: Travel Time pick
- T3: MCCC pick
- T4: Zoom in

## 3.6.4 Post Processing

### 3.6.4.1 Getting the output

In the same folder as the initial PKL file you ran `aimbat-ttpick` on, you can find the output list with extension `<event name>.mcp`, which contains the travel time arrivals.



```
MCCC processed: unknown event at: Fri, 29 Aug 2014 11:42:15 EST
station, mccc delay,    std,    cc coeff,  cc std,   pol   , t0_times  , delay_times
 AR.113A     -1.6201   0.0209   0.9074   0.0765    0  AR.113A.__.BHZ   678.9673     1.6495
 AR.319A     11.6791   0.0361   0.8830   0.0658    0  AR.319A.__.BHZ   691.8687     2.0473
 AR.U15A     12.8982   0.0309   0.8842   0.0857    0  AR.U15A.__.BHZ   693.3771     1.7580
 AR.W13A      4.1460   0.0307   0.8842   0.0904    0  AR.W13A.__.BHZ   684.5807     1.8021
 AR.X16A     11.0675   0.0261   0.8673   0.1030    0  AR.X16A.__.BHZ   691.3650     1.9394
 AR.X18A     16.1568   0.0290   0.8432   0.1088    0  AR.X18A.__.BHZ   696.8047     1.5890
 AZ.BZN     -10.0634   0.0178   0.9072   0.0672    0  AZ.BZN.__.BHZ   670.4364     1.7371
 AZ.CPE     -13.6968   0.0224   0.8904   0.0823    0  AZ.CPE.__.BHZ   667.1692     1.3709
 AZ.CRY     -10.1136   0.0238   0.8979   0.0786    0  AZ.CRY.__.BHZ   670.3904     1.7328
 AZ.FRD      -9.8585   0.0209   0.9107   0.0704    0  AZ.FRD.__.BHZ   670.6781     1.7002
 AZ.HWB     -12.7557   0.0557   0.8321   0.1114    0  AZ.HWB.__.BHZ   668.0825     1.3987
 AZ.KNW      -9.7108   0.0402   0.8884   0.0784    0  AZ.KNW.__.BHZ   670.8972     1.6289
 AZ.LVA2     -9.9828   0.0315   0.8954   0.0814    0  AZ.LVA2.__.BHZ   670.4258     1.8283
 AZ.MONP2   -10.7148   0.0272   0.8906   0.0823    0  AZ.MONP2.__.BHZ   669.6515     1.8706
 AZ.PFO      -9.0966   0.0210   0.9027   0.0763    0  AZ.PFO.__.BHZ   671.5171     1.6232
 AZ.RDM     -10.4369   0.0268   0.8968   0.0821    0  AZ.RDM.__.BHZ   670.1752     1.6248
 AZ.SCI2    -18.6104   0.0516   0.7419   0.0598    0  AZ.SCI2.__.BHZ   662.1949     1.4316
 AZ.SMER    -12.4160   0.0326   0.8998   0.0735    0  AZ.SMER.__.BHZ   668.5347     1.2862
 AZ.SND      -9.6575   0.0122   0.9147   0.0584    0  AZ.SND.__.BHZ   670.7992     1.7802
 AZ.SOL     -14.3948   0.0520   0.8295   0.0768    0  AZ.SOL.__.BHZ   666.5106     1.3314
 AZ.TRO      -9.0127   0.0220   0.9121   0.0658    0  AZ.TRO.__.BHZ   671.3926     1.8316
```

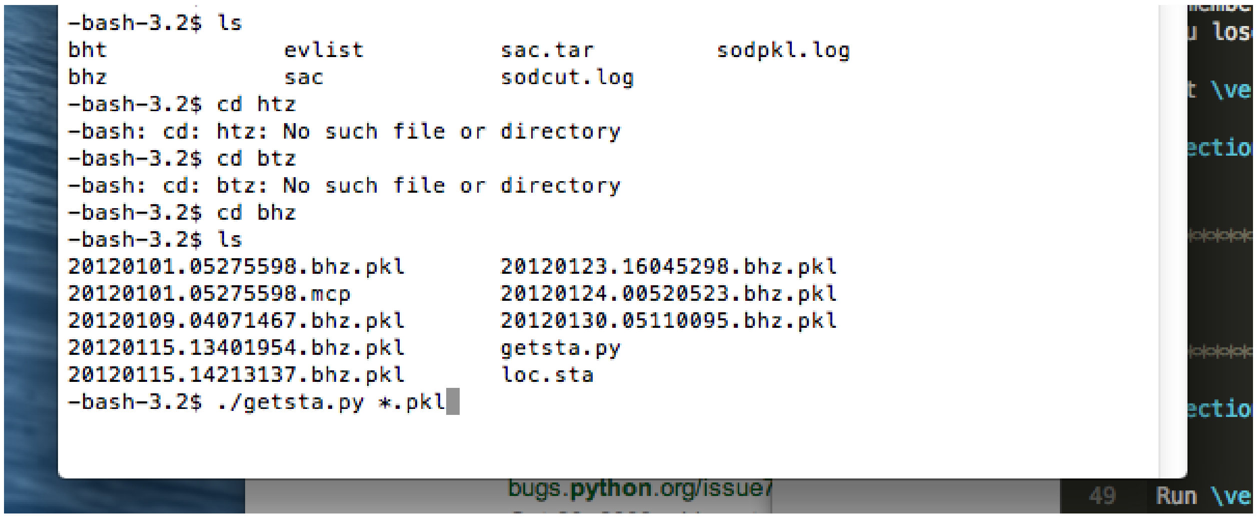
`mccc delay` is *t3+average arrival times*, and *t0_times* are the theoretical arrival times. *delay_times* are obtained by taking *t3-t0*.

#### 3.6.4.2 Disclaimer about delay times

*t0* depends on hypocenter location, origin time, and reference model. We compute the delay time by finding *t3-t0*, but it does not have elliptic, topological, or crust corrections.

#### 3.6.4.3 Getting the stations of the seismograms chosen

Run `getsta.py` in the additional scripts (not on Github for now). It gives the unique list of stations where the seismograms came from. You need to run it with the list of all `pkl` files chosen after you saved to. To do this, type `./getsta.py *.pkl`.



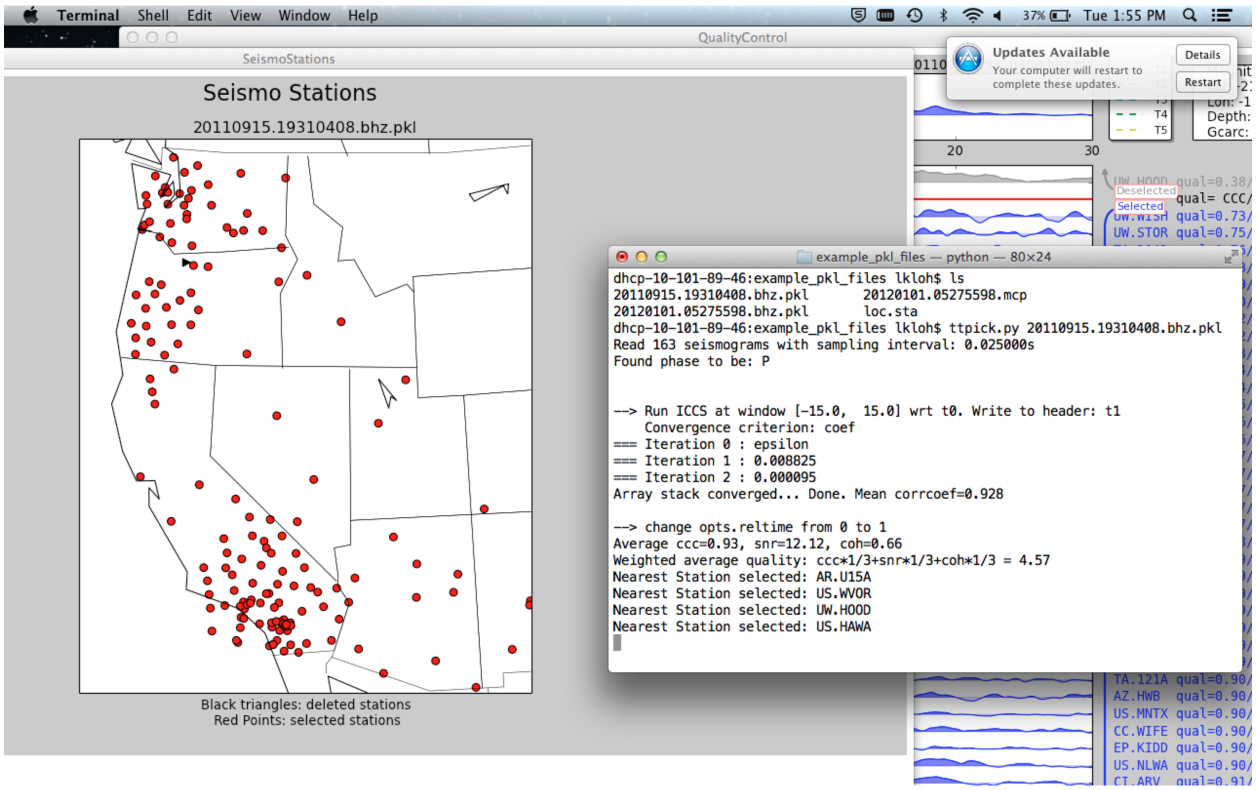
```
-bash-3.2$ ls
bht             evlist          sac.tar         sodpkl.log
bhz             sac             sodcut.log
-bash-3.2$ cd htz
-bash: cd: htz: No such file or directory
-bash-3.2$ cd btz
-bash: cd: btz: No such file or directory
-bash-3.2$ cd bhz
-bash-3.2$ ls
20120101.05275598.bhz.pkl       20120123.16045298.bhz.pkl
20120101.05275598.mcp           20120124.00520523.bhz.pkl
20120109.04071467.bhz.pkl       20120130.05110095.bhz.pkl
20120115.13401954.bhz.pkl       getsta.py
20120115.14213137.bhz.pkl       loc.sta
-bash-3.2$ ./getsta.py *.pkl
```

#### 3.6.4.4 Visualizing Stations on a map

After running:

```
aimbat-ttpick <sac-files>
```

Hit `Map of Stations` in order to get a visual respresentation of where exactly each station is. Dots represent circles used for computing delay times; black triangles represent discarded stations. Click on a dot to get the station name in the terminal.


Terminal
Shell
Edit
View
Window
Help
Tue 1:55 PM
QualityControl
SeismoStations
Seismo Stations
20110915.19310408.bhz.pkl
Black triangles: deleted stations
Red Points: selected stations
Updates Available
Your computer will restart to complete these updates.
Details
Restart
example_pkl_files — python — 80×24
Deselected
Selected

```
dhcp-10-101-89-46:example_pkl_files lkloh$ ls
20110915.19310408.bhz.pkl       20120101.05275598.mcp
20120101.05275598.bhz.pkl       loc.sta
dhcp-10-101-89-46:example_pkl_files lkloh$ ttpick.py 20110915.19310408.bhz.pkl
Read 163 seismograms with sampling interval: 0.025000s
Found phase to be: P


--> Run ICCS at window [-15.0,  15.0] wrt t0. Write to header: t1
    Convergence criterion: coef
=== Iteration 0 : epsilon
=== Iteration 1 : 0.008825
=== Iteration 2 : 0.000095
Array stack converged... Done. Mean corrcoef=0.928

--> change opts.reltime from 0 to 1
Average ccc=0.93, snr=12.12, coh=0.66
Weighted average quality: ccc*1/3+snr*1/3+coh*1/3 = 4.57
Nearest Station selected: AR.U15A
Nearest Station selected: US.WVOR
Nearest Station selected: UW.HOOD
Nearest Station selected: US.HAWA
```

## 3.6.5 Picking Travel Times does not work

If you run `aimbat-ttick <Event name>.bhz.pkl`, a GUI will pop up for you to manually pick the travel times by pressing the keyboard. If typing on the keyboard as directed does not allow you to pick travel times, it could be a problem with the keyboard settings, or the matplotlib backend.

To fix this, first look for the .matplotlib directory. It is hidden in your home directory, so do `ls -a` to find it.

Once you have found the `.matplotlib` directory, cd into it, and then look for the `matplotlibrc` file. Inside that file, ensure the backend is set to:

```
backend : TkAgg
```

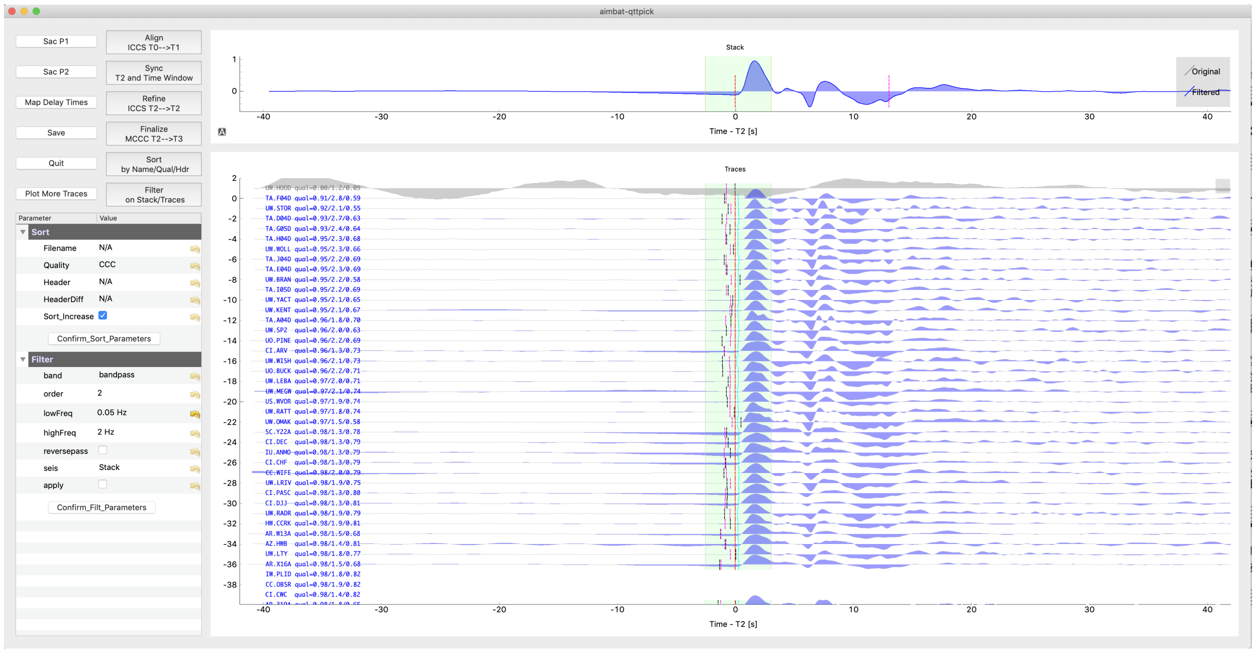
Make sure to comment out the other backends.

## 3.7 Alternative Qt GUI for Measuring Arrival Times

The Matplotlib GUI is slow for interactive plotting. An additional GUI based on pyqtgraph was built since v1.0.0 to speed up plotting. Similar to the old GUI, run:

```
aimbat-qttpick <path-to-pkl-file>
```

to launch the Qt GUI. Phase of the seismogram (P or S), if not given in command line, can be automatically found based on file names including BHZ or BHT. Here is aGn example snapshot:



| Parameter | Value |
|---|---|
| **Sort** | |
| Filename | N/A |
| Quality | CCC |
| Header | N/A |
| HeaderDiff | N/A |
| Sort_Increase | ✓ |
| Confirm_Sort_Parameters | |
| **Filter** | |
| band | bandpass |
| order | 2 |
| lowFreq | 0.05 Hz |
| highFreq | 2 Hz |
| reversepass | ☐ |
| seis | Stack |
| apply | ☐ |
| Confirm_Filt_Parameters | |

The AIMBAT philosophy of using the five-step (`Align`, `Pick`, `Sync`, `Refine`, and `Finalize`) procedure for automated and interactive phase arrival time measurement is the same.

Some GUI behavior remains the same:

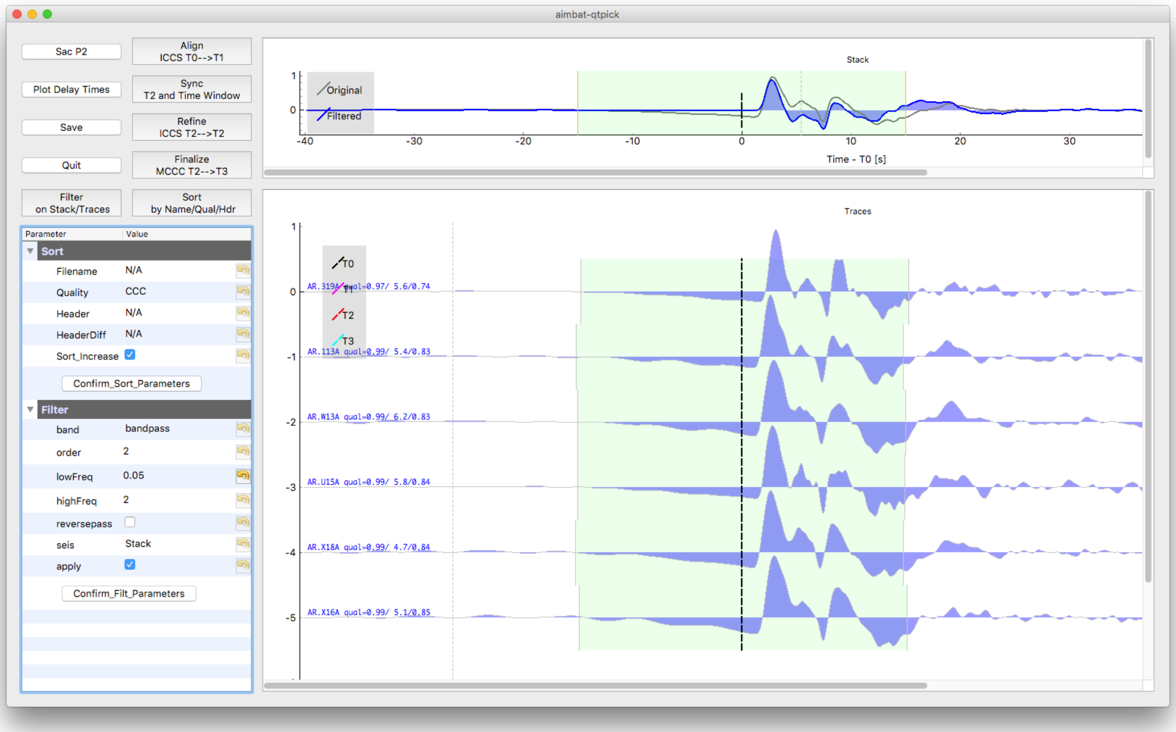
- the phase picking steps of `Align`, `Pick`, `Sync`, `Refine`, and `Finalize`.
- mouse clicking waveforms to change trace selection status.
- keyboard interaction: `t[0-9]` to pick time, and `w` to set time window.
- Click Button *Sac P2* to overlay all traces relative to time picks

Some components are different:

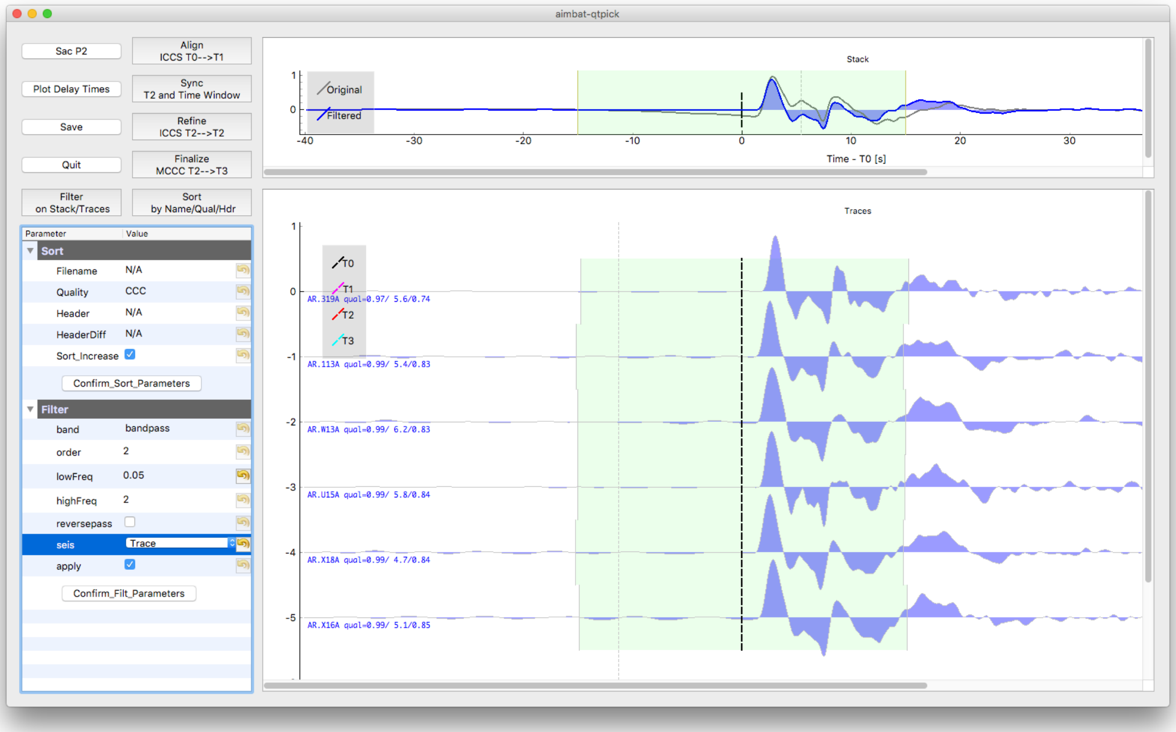
- Choose sort and filter options in a parameter tree and apply in the same GUI window.
- All traces plotted in one long page, instead of multiple pages.
- Still Possible to plot a subset of traces. Click button to add more traces to the plotting window.
- Pyqtgraph mouse events: e.g., right mouse button drag to zoom in and out.
- Time window is plotted as a pyqtgraph.LinearRegionItem instead of Matplotlib Span Selector. To change time window size, just drag either side line and move. Still press key `w` to set time window.
- A vertical hair indicating the time axis value is always plotted follow the mouse movement.

In the above example, 37 selected seismograms are plotted initially. During the arrival time measurement procedure, traces sorting order is changed after time window size or sorting parameters are changed. Trace 37, 38, 39 are missing in the GUI. You can optionally click `Plot More Traces` Button to fill the gap. You can also zoom out vertically and plot more traces.

Here is an example of filtering seismograms. First choose filtering parameters in the parameter tree and test on the stack by clicking Button `Confirm_Filt_Parameters` and Button `Filter on Stack/Traces`. Then applied filter to traces after parameters are finalized.

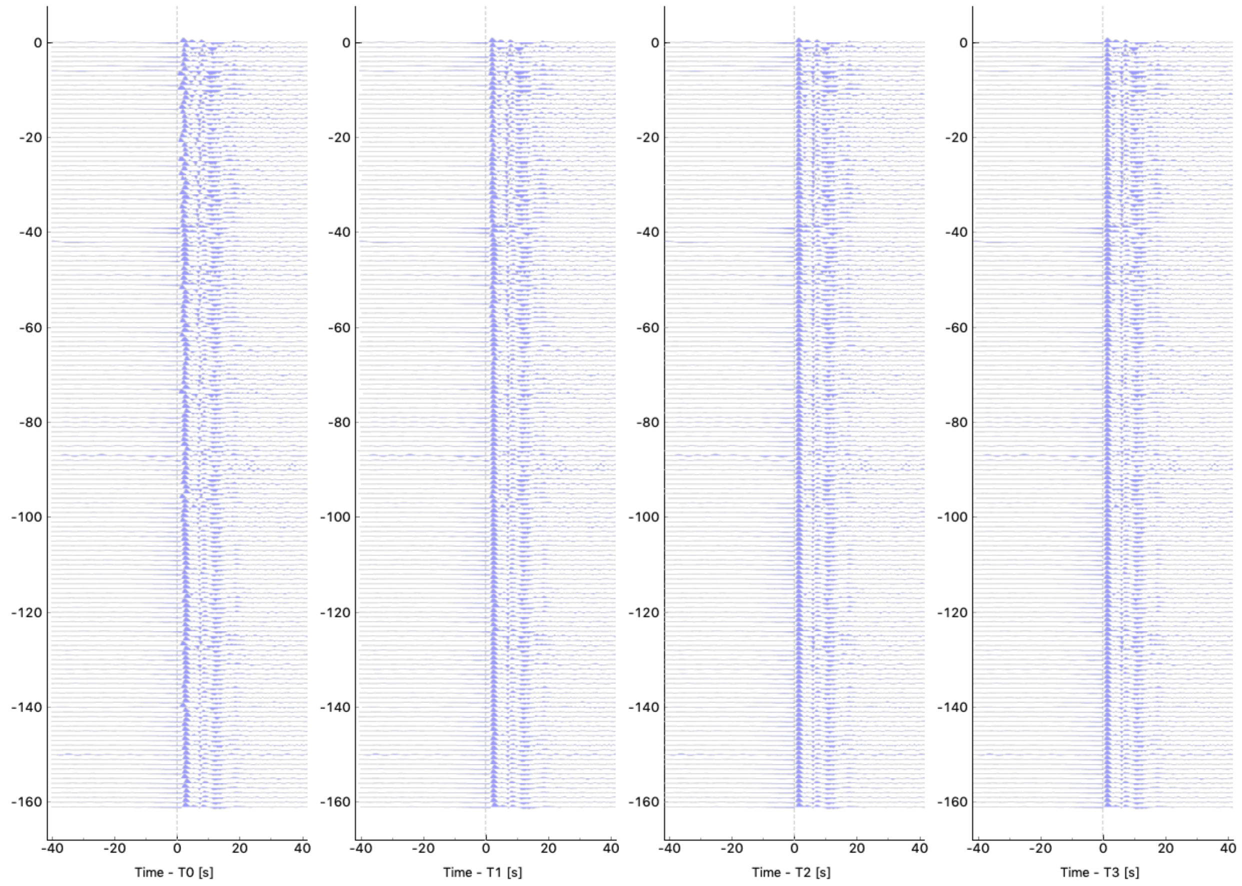


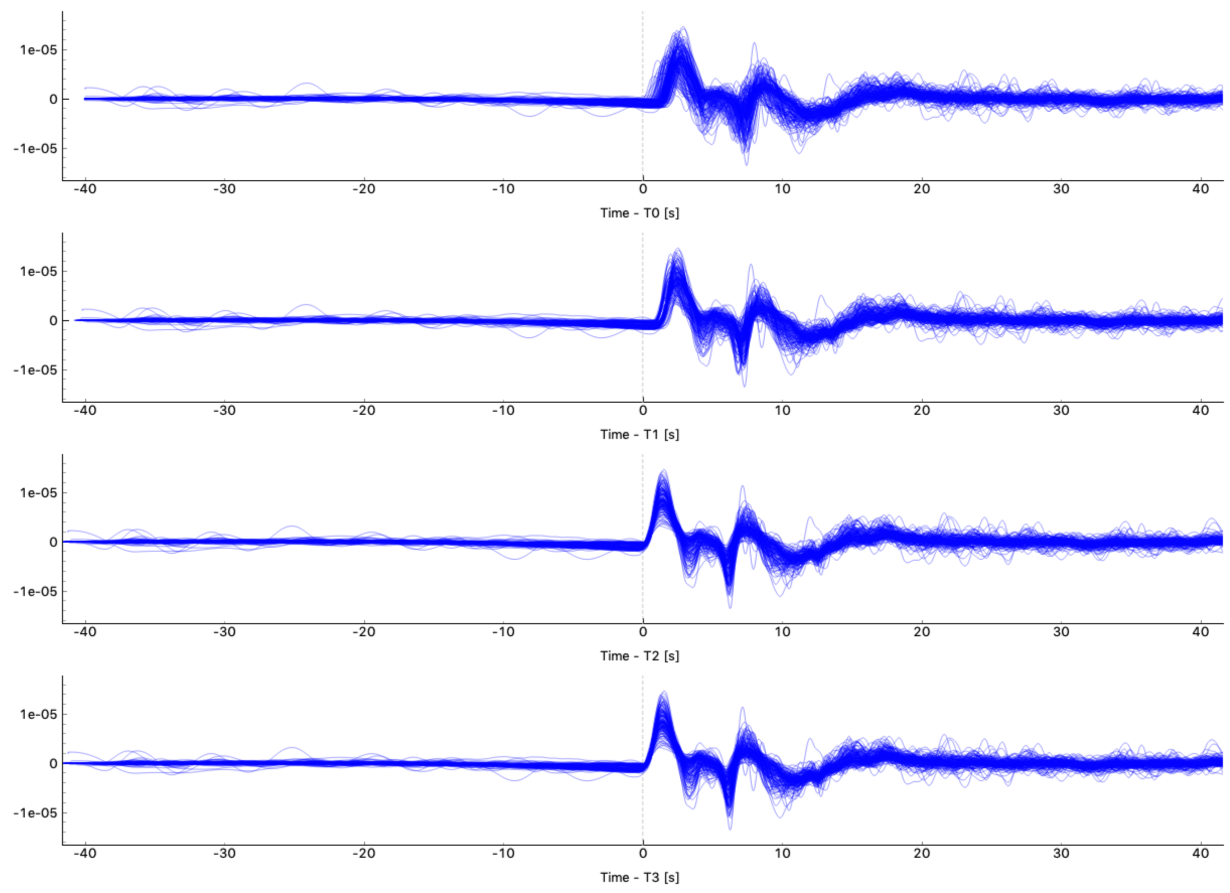
| Parameter | Value |
|---|---|
| Sort | |
| Filename | N/A |
| Quality | CCC |
| Header | N/A |
| HeaderDiff | N/A |
| Sort_Increase | ✓ |
| Filter | |
| band | bandpass |
| order | 2 |
| lowFreq | 0.05 |
| highFreq | 2 |
| reversepass | ☐ |
| seis | Stack |
| apply | ✓ |



| Parameter | Value |
|---|---|
| Sort | |
| Filename | N/A |
| Quality | CCC |
| Header | N/A |
| HeaderDiff | N/A |
| Sort_Increase | ✓ |
| Filter | |
| band | bandpass |
| order | 2 |
| lowFreq | 0.05 |
| highFreq | 2 |
| reversepass | ☐ |
| seis | Trace |
| apply | ✓ |

Some QC tools are available in this Qt GUI. Click Button `Sac P1` and `Sac P2` to plot traces relative to four time picks T0, T1, T2, and T3. Click Button `Plot Delay Times` to plot absolute delay times in a map view.


Time - T0 [s]
Time - T1 [s]
Time - T2 [s]
Time - T3 [s]



images/20110915.19310408.mcp.png

More details might be added here in the future.

CHAPTER

FOUR

# DEVELOPING AIMBAT

For setting up an environment for development of aimbat AIMBAT please follow the instructions on the pysmo development page.

**Caution:** Please adjust the URLs from *github.com:pysmo/pysmo.git* to *github.com:pysmo/aimbat.git*!

CHAPTER

FIVE

# CITATIONS

CHAPTER

SIX

# CREDITS

## 6.1 Lead Developers

- Lay Kuan Loh Email: lloh at ece.cmu.edu
- Xiaoting Lou Email: xlou at u.northwestern.edu
- Suzan van der Lee Email: suzan at earth.northwestern.edu
- Simon Lloyd Email: simon at slloyd.net

## 6.2 Contributers

CHAPTER

# SEVEN

# CHANGELOG

## 7.1 aimbat-v1.0.5

Aug 18, 2019

Main contributor: smlloyd, xlougeo.

- Lots of code clean and restructure, including src, docs, and tests.
- Package installation: make fortran optional, update dependencis, update travis, setup pipenv.
- Update documentation.
- Bug fixes.

## 7.2 aimbat-v1.0.4

Dec 23, 2018

- Add SAC P1 Button to the main GUI
- Plot only a subset of traces for faster data QC and (de)selection. All labels are plotted. Add a button to plot more traces

## 7.3 aimbat-v1.0.3

Dec 7, 2018

- Some GUI setting changes

## 7.4 aimbat-v1.0.2

Dec 5, 2018

- Add option to plot simple delay time map by matplotlib.pyplot

## 7.5 aimbat-v1.0.1

Dec 4, 2018

- Fix bugs in changing trace selection status (QC), manual phase picking, and trace label.
- Change in GUI settings. Using right button dragging is enough and give up on QScrollArea.
- In cross-correlation, do not allow reverse polarity which causes cycle skipping too often.

## 7.6 aimbat-v1.0.0

Dec 3, 2018

Main contributor: smlloyd:

- Use new pysmo.core.sac.SacIO (pysmo-pysmo-v0.7.0) instead of pysmo.sac.sacio.SacFile
- **New setup.py:**
  - Wrap all scripts into a callable function and add them to entry_point, e.g., aimbat-ttpick is automatically generated in your bin folder.
  - Use git commit/tag to determine version automatically.
  - Setup travis
- Package uploaded to pypi.org for each release since this.

Main contributor: xlougeo, ASankaran:

- SAC plotting and aimbat-ttpick are still using Matplotlib GUI.
- **New GUI (aimbat-qttpick) using pyqtgraph for fast plotting. Similar user interactions as v0.3:**
  - Key pressed event handler in pyqtgraph is redefined
  - Use mouse to change time window and press key 'w' to set <– work on stack only
  - Press key 't[0-9]' to set time picks like SAC PPK <– work on both stack and traces
  - Mouse click on waveform to change trace selection status <– work on trace only
- Better separation between data and plot.
- Filter and sort are both in the main GUI controlled by a parameter tree.

## 7.7 aimbat-v0.3-alpha1

June 3, 2018

Main contributor: xlougeo

- Upgrade to python3 (May not back-compatible with python2). No change in functionalities from v0.2.

## 7.8 aimbat-v0.2

Main contributor: lkloh

For changes made between Dec 19, 2012 (v0.1.2) and Aug 23, 2016:

- Added a warning button if you hit ICCS-A or ICCS-B button, to make sure do did not hit it by accident.
- Hitting one of those buttons will undo all the work you did in manually picking arrival times.
- Added a button to allow you to jump to the front page. Note that hitting MCCC again will do just that.
- Added a summary of the event at the top right hand corner: Magnitude, Location, Depth
- Added a GUI to allow sorting of the seismograms according to header, time difference, file name, …
- Added a button to return to original screen after you zoom in/out

## 7.9 aimbat-0.1.2

Dec 19, 2012

Main contributor: xlougeo

- Change sci format for scientific notation of sacp2: from 1e-5 to 10^{-5}
- Change font properties for station label to "monospace" for equal width
- Minor changes in program descriptions, example scripts
- Further code development on github.com after this version.

## 7.10 aimbat-0.1.1

Sep 27, 2012

Main contributor: xlougeo

- Change setup.py and package directory: modules –> src/pysmo/aimbat.
- AIMBAT becomes a part of pysmo (https://github.com/pysmo/aimbat).
- Python usage: import aimbat –> from pysmo import aimbat
- Minor changes in help messages for scripts using the OptionParser module.
- Adjust figsize-related function of ttpick.py to support backends other than Tk.

## 7.11 aimbat-0.1

Sep 19, 2012

Main contributor: xlougeo

First release on Northwestern website

CHAPTER

EIGHT

# INDICES AND TABLES

- genindex
- modindex
- search

# BIBLIOGRAPHY

[GoldsteinDodge2003] 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.

[Hunter2007] Hunter, J. (2007), Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering, 3(9), 90–95.

[LouVanDerLeeLloyd2013] AIMBAT: A Python/Matplotlib Tool for Measuring Teleseismic Arrival Times. Xiaoting Lou, Suzan van der Lee, and Simon Lloyd (2013), Seismol. Res. Lett., 84(1), 85-93, doi:10.1785/0220120033.

[VanDecarCrosson1990] 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.

[BulandChapman1983] Ray Buland and C. H. Chapman (1983), The Computation of Seismic Travel Times, Bulletin of the Seismological Society of America, 73(5), 1271-1302.