
AIMBAT Documentation

Release 0.2.1

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

July 06, 2015

CONTENTS

| | | |
|----------|---|-----------|
| 1 | Introduction | 3 |
| 1.1 | About AIMBAT | 3 |
| 1.2 | Associated Documents | 3 |
| 1.3 | Authors' Contacts | 3 |
| 2 | Installing Dependencies | 5 |
| 2.1 | Getting your operating system | 5 |
| 2.2 | Github | 6 |
| 2.3 | MacPorts | 7 |
| 2.4 | Python and its Dependencies | 7 |
| 2.5 | Installing Basemap without MacPorts | 9 |
| 2.6 | Fortran | 9 |
| 2.7 | Possible Issues | 10 |
| 3 | Installing AIMBAT | 15 |
| 3.1 | Getting the Packages | 15 |
| 3.2 | Building the Pysmo Packages | 15 |
| 3.3 | Example Data | 16 |
| 4 | Upgrading AIMBAT | 17 |
| 4.1 | Getting the latest version | 17 |
| 4.2 | Possible Issues | 18 |
| 5 | Getting Data | 19 |
| 5.1 | Obspy.fdsn for downloading data | 19 |
| 5.2 | Standing Order for Data | 19 |
| 6 | SAC Input/Output procedures for AIMBAT | 23 |
| 6.1 | Converting from SAC to PKL files | 23 |
| 6.2 | Converting from PKL to SAC files | 23 |
| 7 | Filtering Data | 25 |
| 7.1 | Filtering GUI | 25 |
| 7.2 | Saving Options | 25 |
| 8 | Analyzing Data | 27 |
| 8.1 | Seismic Analysis Code (SAC) | 27 |
| 9 | Parameter Configuration | 29 |
| 9.1 | Backend | 29 |
| 9.2 | Configuration File | 29 |

| | |
|---|-----------|
| 10 SAC Data Access | 33 |
| 10.1 Python Object for SAC File | 33 |
| 10.2 Python Pickle for SAC Files | 35 |
| 10.3 SAC Plotting and Phase Picking | 36 |
| 10.4 SAC Phase Picking | 41 |
| 11 Measuring Teleseismic Body Wave Arrival Times | 45 |
| 11.1 Automated Phase Alignment | 45 |
| 11.2 Picking Travel Times | 46 |
| 11.3 What the Alignments Stand For | 53 |
| 11.4 Post Processing | 54 |
| 12 Visualizing Stations on a map | 57 |
| 13 Unit Testing | 59 |
| 13.1 Running the Tests | 59 |
| 14 Updating this manual | 61 |
| 14.1 Dependencies | 61 |
| 14.2 How to update this manual | 61 |
| 15 Citations | 63 |
| 16 Indices and tables | 65 |
| Bibliography | 67 |

Contents:



NORTHWESTERN UNIVERSITY

INTRODUCTION

1.1 About AIMBAT

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 [[LouVanDerLee2013](#)]. 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 programming language 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 16.

The AIMBAT software package is distributed under the GNU General Public License Version 3 (GPLv3) as published by the Free Software Foundation.

1.2 Associated Documents

- Seismological Research Letters Paper
- PDF Version of Manual. Automatically generated from these online docs; please excuse minor issues that may arise from automated conversion.

1.3 Authors' Contacts

- Lay Kuan Loh
Email: llloh at ece.cmu.edu
- Xiaoting Lou
Email: xlou at u.northwestern.edu
- Suzan van der Lee
Email: suzan at earth.northwestern.edu

INSTALLING DEPENDENCIES

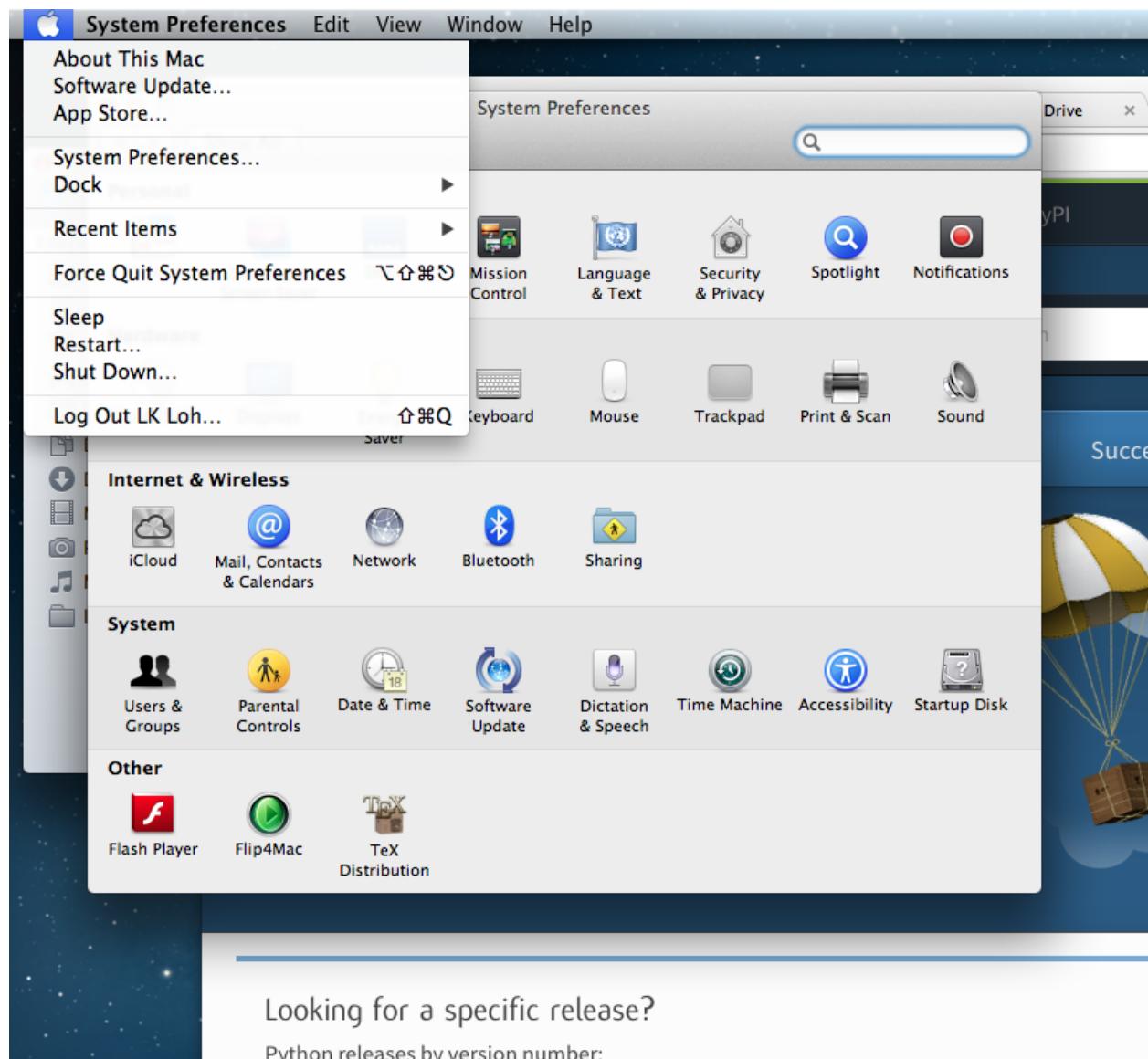
AIMBAT is built on top of [standard Python](#), and uses several extra Python libraries as well. This page guides the user through checking if the required libraries are already installed on the computer they want to install AIMBAT on, and how to get them if not.

To install AIMBAT, you will need sudo/root privileges for the computer you are installing it on.

2.1 Getting your operating system

We assume that most users of AIMBAT will be using Macs. If our assumptions are wrong, please [contact the authors](#), and if there is sufficient interest we will test and write documentation for installations on other operating systems as well.

On a Mac, to find the version of your operating system, first click on the Apple icon on the top bar of your desktop, go to System Preferences, and click on Startup Disk. The operating system version should then be displayed.



2.2 Github

Optional.

The latest version of AIMBAT will be on [Github](#), so it would be good to get GitHub on your computer. This is not strictly necessary, as you could also download it as a zipfile from the [AIMBAT](#) website.

To check if you already have [Github](#) installed, open the terminal and type:

```
git --version
```

If GitHub is installed, the terminal should output a line describing the version number of GitHub installed on the computer, such as:

```
git version 1.8.5.2 (Apple Git-48)
```

If GitHub is not installed, the terminal will respond by telling you that it is not possible to issue this command:

```
-bash: git: command not found
```

To install GitHub, download the package installer [here](#). This would allow only command line usage of Git, so if you want to use a GUI, we recommend [Git for Mac](#).

2.3 MacPorts

[MacPorts](#) is a package management system that will be very helpful to install some python libraries.

To check if MacPorts is installed already, in the terminal, type:

```
port version
```

If MacPorts is installed, you should see the terminal output the version number, for instance:

```
Version: 2.3.0
```

If MacPorts is not installed, you should see the terminal output that `port` is not a valid command:

```
-bash: port: command not found
```

To get MacPorts, download the package installer [here](#) and follow the instructions to install it. Be sure to get the right version of MacPorts for your operating system.

2.4 Python and its Dependencies

AIMBAT requires python 2.7 and above to run. Note that Python is usually installed by default on Mac computers.

AIMBAT requires the following packages to run:

1. [Numpy](#): Used for manipulating numbers and datasets
2. [Scipy](#): Used for data processing
3. [Matplotlib](#): Used for the majority of the plots in AIMBAT and the GUI
4. [Basemap](#): Used for plotting world maps

2.4.1 Checking if Python is installed

Open the terminal and type:

```
python --version
```

If Python is installed, the terminal will output the version number installed, for example:

```
Python 2.7.8
```

If Python is not installed, the terminal will output:

```
-bash: python: command not found
```

2.4.2 If Python is not installed

Inside the terminal, once python is installed, type these commands in using sudo mode. Note you will need to enter your admin password.:

```
sudo port install py27
sudo port install py27-numpy
sudo port install py27-scipy
sudo port install py27-matplotlib
sudo port install py27-matplotlib-basemap
sudo port install py27-ipython
sudo port install python_select
```

Installing the last two packages is 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
```

2.4.3 If Python is already installed

If Python is already installed, first check if you have the four required dependencies. Open up the Python console by typing:

```
python
```

in the terminal. You should see something like this as output:

```
Python 2.7.8 (default, Oct  3 2014, 02:34:26)
[GCC 4.2.1 Compatible Apple LLVM 5.1 (clang-503.0.40)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Now, check if the packages have been installed properly by typing the following in:

```
import numpy
import scipy
import matplotlib
from mpl_toolkits.basemap import Basemap
```

If any of the packages are missing (e.g. `scipy` not installed), the python console will output an error, for instance:

```
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
ImportError: No module named scipy
```

Otherwise, the python console will simply show that it is ready for the next command.

If any of the packages are missing, you can choose to install it by whatever means you are most comfortable with. We provide one possible way to do so using MacPorts below. In the terminal, type:

```
sudo port install py27
```

to get the python version installed in `opt/local/bin` where MacPorts installs everything to. Select to use this version of Python by typing:

```
sudo port install python_select
```

Now, install the missing packages by doing:

```
sudo port install py27-numpy
sudo port install py27-scipy
sudo port install py27-matplotlib
sudo port install py27-matplotlib-basemap
```

2.5 Installing Basemap without MacPorts

If you have already installed Basemap, which means that:

```
from mpl_toolkits.basemap import Basemap
```

comes out without an error in the Python console, you can skip this section. This is for users who do not want to use the MacPorts version of Python which has been installed to /opt/local/bin. We anticipate that users who installed the official version of Python from the [Python website](#) may possibly find this section useful.

Disclaimer: Lifted from content written by [this guy](#) with some tweaks.

Enthought Python (aka Canopy) should get you most of the dependencies needed, and can be downloaded [here](#). You do need to get [Geos](#) though. The best way to get it is [install Homebrew](#), and then install gdal, a package that has Geos as a dependency. To get gdal, do:

```
brew install gdal
```

Now install Basemap. Download it [here](#). Unzip the package and cd into the unzipped package. To install basemap, do:

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

To check it worked, at the terminal, do:

```
python
```

and then:

```
from mpl_toolkits.basemap import Basemap
```

If installed correctly, no errors will occur.

2.6 Fortran

GFortran must be installed for AIMBAT. Note that f90 or f77 are also acceptable. To check if you already have GFortran, in the terminal type:

```
gfortran --version
```

which will print out the version number you have, if it is already installed.

If GFortran is not installed, the terminal will respond with:

```
-bash: gfortran: command not found
```

GFortran is available [here](#). Be sure to get the correct version for your operating system.

2.7 Possible Issues

Here some common problems and possible resolutions. If your problem is not listed here, or you have a suggestion, please [contact the authors](#).

2.7.1 Macports

Many issues encountered can be solved by upgrading all Python directories on your computer using MacPorts. In the command line type:

```
sudo port selfupdate  
sudo port upgrade outdated
```

This will first make sure your version of MacPorts is the most recent version, and will then go through all Python libraries to make sure they are all upgraded as well. Be aware that this step can take upwards of an hour depending on how many Python libraries you have installed. If at any point a library fails to install, type:

```
sudo port clean <library>  
sudo port install <library>
```

For example, if the library libcaca outputs an error, type:

```
sudo port clean libcaca  
sudo port install libcaca
```

You can then simply retype `sudo port upgrade outdated` and it will continue with the upgrade where it left off.

If this last step does not work (i.e. it still outputs an error), type `sudo port -p upgrade outdated`, which will ignore any errors the upgrade runs into and simply continue through the process. It is likely that these ignored errors will not have any effect on AIMBAT itself.

You may also run into problems with AIMBAT if your [Macport](#) version is not compatible with your operating system version. For example, if you used Macports for OS X 10.8 to install AIMBAT, then upgraded your operating system or OS X 10.9, you may find that AIMBAT no longer works properly. You will need to upgrade Macports to fix this error.

Do not uninstall MacPorts unless you know what you are doing, uninstalling MacPorts may get rid of other programs you installed using MacPorts. However, if you are sure you want to do so, see [here](#) for instructions.

2.7.2 Installing Python with Pip

Be careful with the operating system. For OS X 10.9 and above, Python 2.7 is not fully compatible and there may be problems installing python with Pip. Best to use Enthought Canopy or Python 3 with OS X 10.9.

2.7.3 Setting the Python Path to the scripts

You are asked to add the path to the AIMBAT scripts in your file. To do that, you add them to the `.bashrc` file. There are other files you could add it to that work as well, such as the `.profile` or `.bash_profile` files. You can see the files by opening the terminal, going to your home directory and doing `ls -a` to see all the hidden files, and open then by doing `vi .bashrc` in vim, for instance. If the `.bashrc` file does not exist, `vi .bashrc` will create it and open it for editing. To ensure you can open a script, you need to add:

```
export PATH=$PATH:<path-to-folder-with-scripts>
export PYTHONPATH=$PYTHONPATH:<path-to-folder-with-scripts>
```

to the `.bashrc` file. We recommend adding the paths to the `.bashrc` file.

2.7.4 Terminal Commands stop working

If ever the terminal commands such as `ls` stop working in the terminal, it could be that something went wrong with a path in the `.bashrc` or `.profile` files. If that happens you may not be able to open them in vim as that command would have stopped working as well. Instead, in the terminal, you do:

```
PATH=/bin:${PATH}
PATH=/usr/bin:${PATH}
```

And that should allow the commands to start working again. Figure out what you did wrong and remove that command.

2.7.5 Path to python files not found

After adding the path to your directory with scripts in `.bashrc`, you still need to source the `.bashrc` files in `.profile`, or the system may not find the directory. See here for more [details](#) to see how the profile file is sourced. Note that this one will override the file in `/etc/profile`.

```
earthcomputer — bash — 80x24
dhcp-10-101-142-84:~ earthcomputer$ ls
Desktop      Library      Pictures      pymso
Documents    Movies       Public       tmp
Downloads   Music        aimbat-install
dhcp-10-101-142-84:~ earthcomputer$ python
-bash: /Users/earthcomputer/Library/Enthought/Canopy_64bit/User/bin/python: No such file or directory
dhcp-10-101-142-84:~ earthcomputer$
```

topics

AIX documentation

.profile file

The **.profile** file is present in your home (**\$HOME**) directory and lets you customize your individual working environment.

Because the **.profile** file is hidden, use the **ls -a** command to list it.

After the **login** program adds the **LOGNAME** (login name) and **HOME** (login directory) variables to the environment, the commands in the **\$HOME/.profile** file are executed if the file is present. The **.profile** file contains your individual profile that overrides the variables set in the **/etc/profile** file. The **.profile** file is often used to set exported environment variables and terminal modes. You can customize your environment by modifying the **.profile** file. Use the **.profile** file to control the following defaults:

- Shells to open
- Prompt appearance
- Keyboard sound

The following example is a typical **.profile** file:

```
PATH=/usr/bin:/etc:/home/bin1:/usr/lpp/tps4.0/user::  
epath=/home/gsc/e3:  
export PATH epath  
csh
```

This example has defined two path variables (**PATH** and **epath**), exported them, and opened a C shell (**csh**).

You can also use the **.profile** file (or if it is not present, the **/etc/profile** file) to determine login shell variables. You can also customize other shell environments. For example, use the **.cshrc** file and **.kshrc** file to customize a C shell and a Korn shell, respectively, when each type of shell is started.

Parent topic: [System startup files](#) [[Feedback](#)]

This explanation explains how the **bashrc** file is sourced.

searches the directories in **PATH** for the script.

Invocation

A **login shell** is one whose first character of argument zero is a **-**, or one started with the **--login** option.

An **interactive shell** is one started without non-option arguments and without the **-c** option whose standard input and error are both connected to terminals (as determined by **isatty(3)**), or one started with the **-i** option. **PS1** is set and **\$-** includes **i** if **bash** is interactive, allowing a shell script or a startup file to test this state.

The following paragraphs describe how **bash** executes its startup files. If any of the files exist but cannot be read, **bash** reports an error. Tildes are expanded in file names as described below under **Tilde Expansion** in the **EXPANSION** section.

When **bash** is invoked as an interactive login shell, or as a non-interactive shell with the **--login** option, it first reads and executes commands from the file **/etc/profile**, if that file exists. After reading that file, it looks for **~/.bash_profile**, **~/.bash_login**, and **~/.profile**, in that order, and reads and executes commands from the first one that exists and is readable. The **--noprofile** option may be used when the shell is started to inhibit this behavior.

When a login shell exits, **bash** reads and executes commands from the files **~/.bash_logout** and **/etc/bash.bash_logout**, if the files exist.

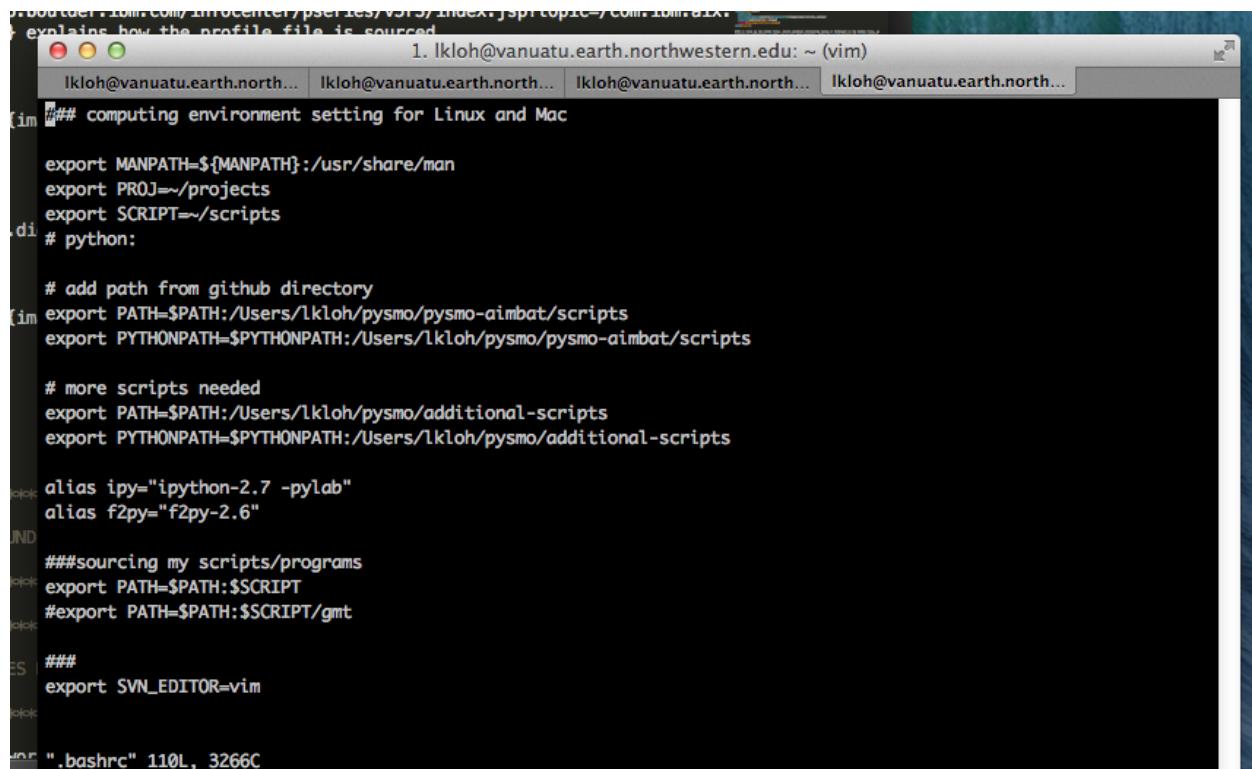
When an interactive shell that is not a login shell is started, **bash** reads and executes commands from **~/.bashrc**, if that file exists. This may be inhibited by using the **--norc** option. The **--rcfile** file option will force **bash** to read and execute commands from **file** instead of **~/.bashrc**.

When **bash** is started non-interactively, to run a shell script, for example, it looks for the variable **BASH_ENV** in the environment, expands its value if it appears there, and uses the expanded value as the name of a file to read and execute. **Bash** behaves as if the following command were executed:

```
if [ -n "$BASH_ENV" ]; then . "$BASH_ENV"; fi  
but the value of the PATH variable is not used to search for the file name.
```

If **bash** is invoked with the name **sh**, it tries to mimic the startup behavior of historical versions of **sh** as closely as possible, while conforming to the **POSIX** standard as well. When invoked as an interactive login shell, or a non-interactive shell with the **--login** option, it first attempts to read and execute

This is what the **bashrc** and **profile** files should look like on your home directory:



The screenshot shows a terminal window titled "1. lkloh@vanuatu.earth.northwestern.edu: ~ (vim)". The window displays the contents of the .bashrc file. The code includes environment variable exports for MANPATH, PROJ, and SCRIPT, as well as Python paths and aliases for ipy and f2py. It also contains sections for sourcing additional scripts and setting the SVN_EDITOR. The bottom status bar indicates 110L, 3266C.

```

#!/bin/bash
# explains how the profile file is sourced
1. lkloh@vanuatu.earth.northwestern.edu: ~ (vim)
lkloh@vanuatu.earth.north... lkloh@vanuatu.earth.north... lkloh@vanuatu.earth.north... lkloh@vanuatu.earth.north...
{im ## computing environment setting for Linux and Mac

export MANPATH=${MANPATH}:/usr/share/man
export PROJ=~/projects
export SCRIPT=~/scripts
di # python:

# add path from github directory
im export PATH=$PATH:/Users/lkloh/pysmo/pysmo-aimbat/scripts
export PYTHONPATH=$PYTHONPATH:/Users/lkloh/pysmo/pysmo-aimbat/scripts

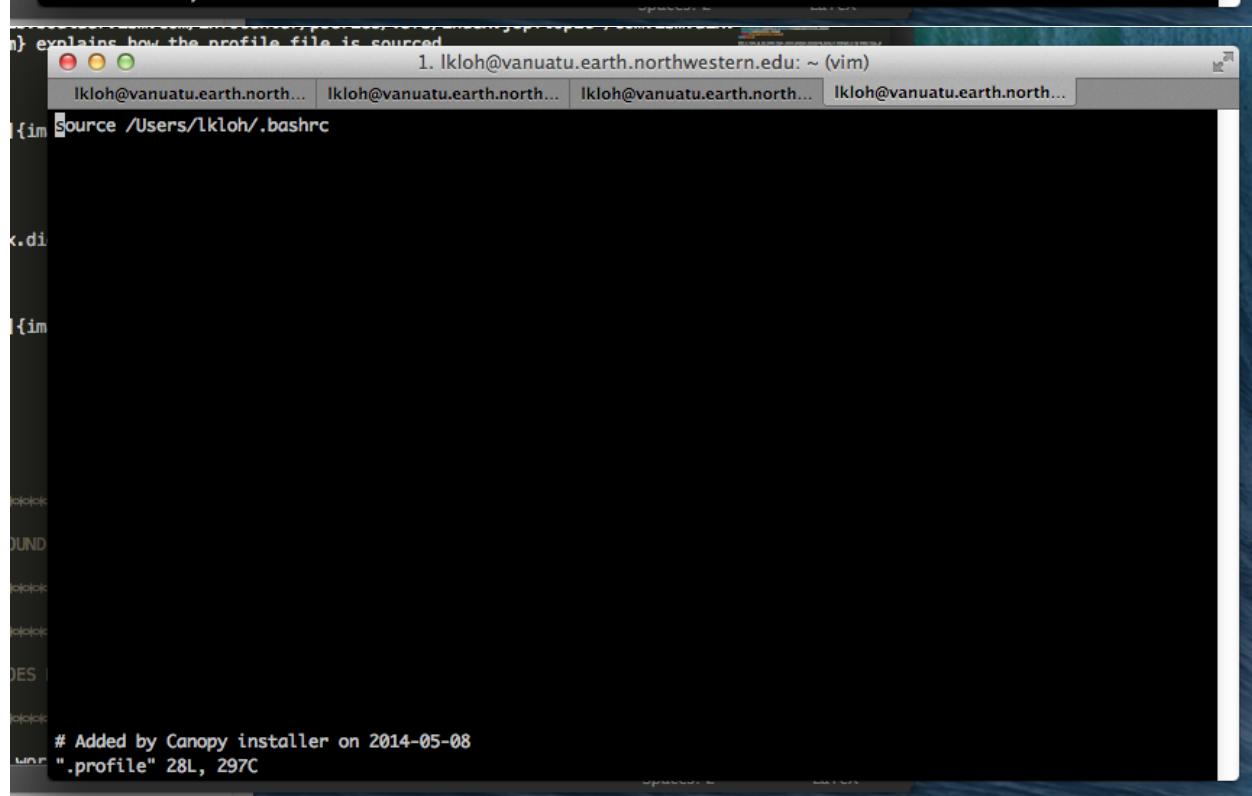
# more scripts needed
export PATH=$PATH:/Users/lkloh/pysmo/additional-scripts
export PYTHONPATH=$PYTHONPATH:/Users/lkloh/pysmo/additional-scripts

alias ipy="ipython-2.7 -pylab"
alias f2py="f2py-2.6"
UND
###sourcing my scripts/programs
export PATH=$PATH:$SCRIPT
#export PATH=$PATH:$SCRIPT/gmt
DES

####
export SVN_EDITOR=vim
ES

".bashrc" 110L, 3266C

```

The screenshot shows a terminal window titled "1. lkloh@vanuatu.earth.northwestern.edu: ~ (vim)". The window displays the contents of the .profile file. It includes a comment about being added by Canopy installer on 2014-05-08. The bottom status bar indicates 28L, 297C.

```

# Added by Canopy installer on 2014-05-08
".profile" 28L, 297C

```


INSTALLING AIMBAT

3.1 Getting the Packages

AIMBAT is released as a sub-package of psmo in the name of `pysmo.aimbat` along with another sub-package `pysmo.sac`. The latest stable release of AIMBAT is available for download at the [official project webpage](#).

We are working on a new release of AIMBAT, available on [Github](#). We are adding more features to make using AIMBAT more convenient, and fixing some bugs in the old code. Download `pysmo.aimbat` and `pysmo.sac` from Github. You will now have two folders called `aimbat` and `sac` respectively.

You may want to download [example code](#) to run AIMBAT on, as well.

3.2 Building the Psmo Packages

You need to be an administrator on the computer you are installing AIMBAT on, as you need to run the commands with `sudo`.

3.2.1 Building `pysmo.sac`

Python module `Distutils` is used to write a `setup.py` script to build, distribute, and install `pysmo.sac`. `cd` into the `sac` directory on the command line and run:

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

If you successfully installed the `sac` module, in the python console, after you type:

```
from psmo import sac
```

there should be no errors.

3.2.2 Installing `pysmo.aimbat`

Three sub-directories are included in the `aimbat` directory:

- `example`: Example SAC files
- `scripts`: Python scripts to run at the command line
- `src`: Python modules to install

The core cross-correlation functions 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 Disutils.

Note that some sort of Fortran compiler must already be installed first. Specify them in place of `gfortran` in the following commands.

cd into the `aimbat` directory and run:

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

to install the `src` directory.

Add `<path-to-folder>/aimbat/scripts` to environment variable `PATH` in a shells start-up file for command line execution of the scripts. Inside the `.bashrc` file, add the lines

Bash Shell Users:

```
export PATH=$PATH:<path-to-folder>/aimbat/scripts
```

C Shell Users:

```
setenv PATH=$PATH:<path-to-folder>/aimbat/scripts
```

If AIMBAT has been installed, type `from pysmo import aimbat` in a Python shell, and no errors should appear.

If you have added the scripts correctly, typing part of the name of the script in the terminal should be sufficient to allow the system to autocomplete the name.

3.3 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.

UPGRADING AIMBAT

4.1 Getting the latest version

The latest version of AIMBAT, currently version 0.1.3, is hosted on the [pysmo](#) repository on Github. We will periodically be making updates to it.

To upgrade AIMBAT, first delete the old AIMBAT files on your computer.

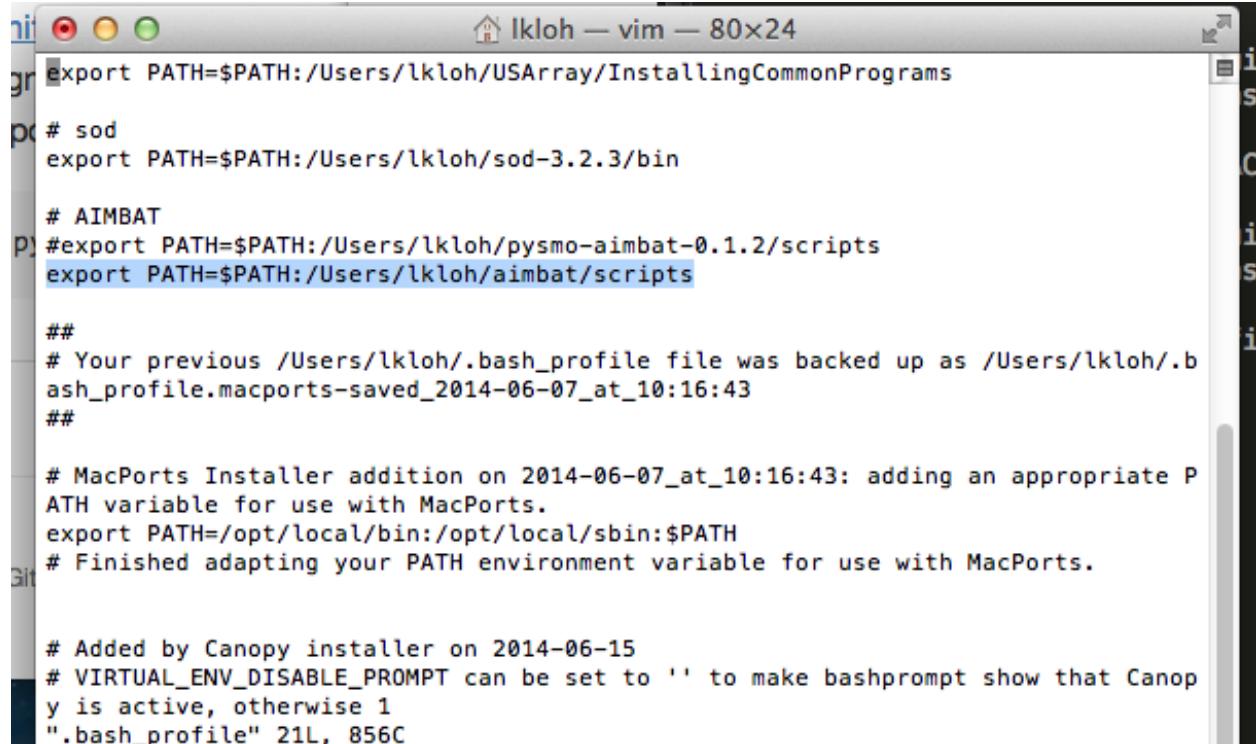
Next, download the newest version of [AIMBAT](#) and [SAC](#) from github. Now, cd into the newest AIMBAT folder and run:

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

Also, cd into the newest SAC folder and run:

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

Now, go to the *.profile* file and add the *scripts* folder in the new AIMBAT version to your path.



```
Ikloh — vim — 80x24
export PATH=$PATH:/Users/lkloh/USArray/InstallingCommonPrograms

# sod
export PATH=$PATH:/Users/lkloh/sod-3.2.3/bin

# AIMBAT
#export PATH=$PATH:/Users/lkloh/pysmo-aimbat-0.1.2/scripts
export PATH=$PATH:/Users/lkloh/aimbat/scripts

##
# Your previous /Users/lkloh/.bash_profile file was backed up as /Users/lkloh/.bash_profile.macports-saved_2014-06-07_at_10:16:43
##

# MacPorts Installer addition on 2014-06-07_at_10:16:43: adding an appropriate PATH variable for use with MacPorts.
export PATH=/opt/local/bin:/opt/local/sbin:$PATH
# Finished adapting your PATH environment variable for use with MacPorts.

# Added by Canopy installer on 2014-06-15
# VIRTUAL_ENV_DISABLE_PROMPT can be set to '' to make bashprompt show that Canopy is active, otherwise 1
".bash_profile" 21L, 856C
```

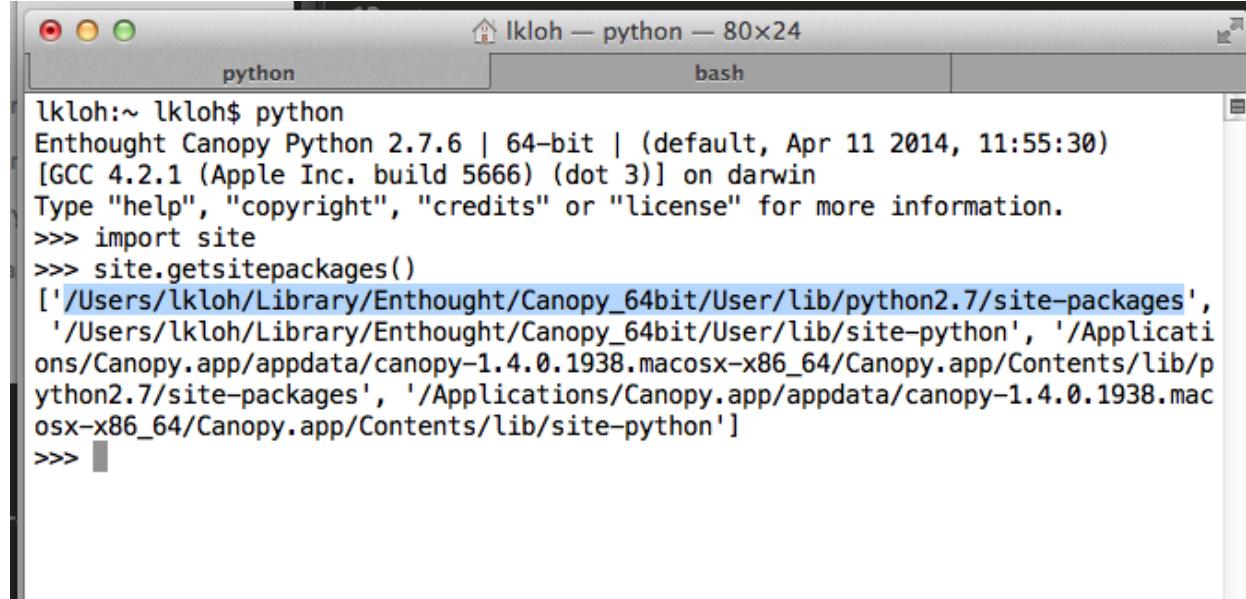
4.2 Possible Issues

Some users have reported errors with upgrading. If there are any problems running the updated AIMBAT, try the following:

Get the location of the python site-packages directory by typing the following into the python console:

```
import site  
site.getsitepackages()
```

The path to the site packages directory is highlighted in the figure below.



```
lkloh:~ lkloh$ python  
Enthought Canopy Python 2.7.6 | 64-bit | (default, Apr 11 2014, 11:55:30)  
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin  
Type "help", "copyright", "credits" or "license" for more information.  
>>> import site  
>>> site.getsitepackages()  
['/Users/lkloh/Library/Enthought/Canopy_64bit/User/lib/python2.7/site-packages',  
 '/Users/lkloh/Library/Enthought/Canopy_64bit/User/lib/site-python', '/Applications/Canopy.app/appdata/canopy-1.4.0.1938.macosx-x86_64/Canopy.app/Contents/lib/python2.7/site-packages', '/Applications/Canopy.app/appdata/canopy-1.4.0.1938.macosx-x86_64/Canopy.app/Contents/lib/site-python']  
>>>
```

cd into the site-packages directory and remove all the folders with the word `pysmo` in their names by typing:

```
sudo rm -rf <pysmo-folder>
```

Now, reinstall the new version of AIMBAT.

If this still does not work, upgrade your Python libraries as detailed in section 2.7.1 of this manual and reinstall the new version of AIMBAT.

GETTING DATA

Note: Not necessary if you already have your own set ways of obtaining data. This section is added for completeness. There are several ways to obtain seismic data from [IRIS](#) to input into AIMBAT. The authors used two ways to do it, and a further list of libraries for obtaining seismic data is provided in the sidebars [here](#).

5.1 Obspy.fdsn for downloading data

5.1.1 Installing Obspy

We recommend using Macports to install Obspy as detailed in the *Installation* section [here](#). If you have installed Enthought Canopy:

```
sudo port install py27-obspy
```

should install Obspy. If not, installing it with Homebrew also seems to work.

5.1.2 Did the installation work?

If the installation has worked, *close the terminal* you used to install Obspy on, and then open it again. Now, open the Python terminal in a new terminal by typing `python`, and type:

```
import obspy
```

If there are no errors, your installation has worked.

5.1.3 Using Obspy

Use the [Obspy FDSN](#) web service client for Obspy in Python. Once you have done so, check out the [SAC-Input Output](#) libraries for loading the data to Python and saving it as SAC or Pickle files.

5.2 Standing Order for Data

Note: NOT needed for AIMBAT, but important to know about as it is a commonly used package for downloading seismic data with the user's specifications. Although Obspy also offers ways to download seismic data from IRIS, SOD allows for better fine-tuning of obtained data.

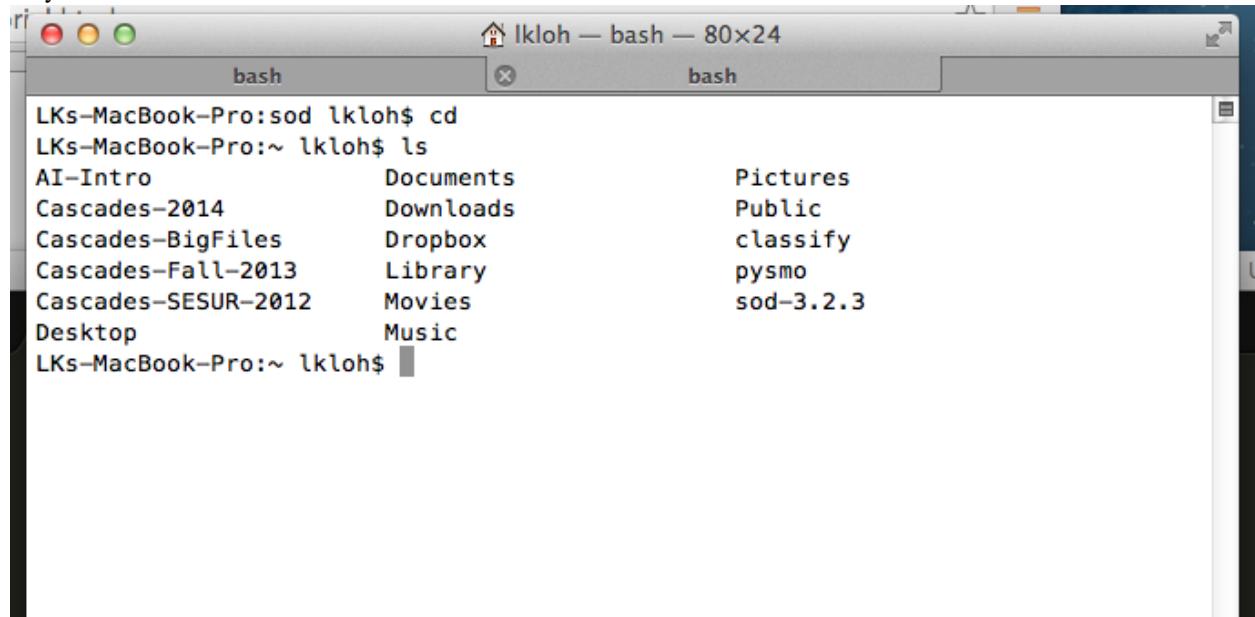
From the [SOD](#) website:

Standing Order for Data is a framework to define rules to select seismic events, stations, and data. It then allows you to apply processing to the events, stations, and data and currently contains a large set of rules that allow you to select with great precision in these items. The processes mainly consist of simple data transformation and retrieval, but SOD defines hooks to allow you to cleanly insert your own processing steps, either written in Java or an external program.

5.2.1 Installing SOD

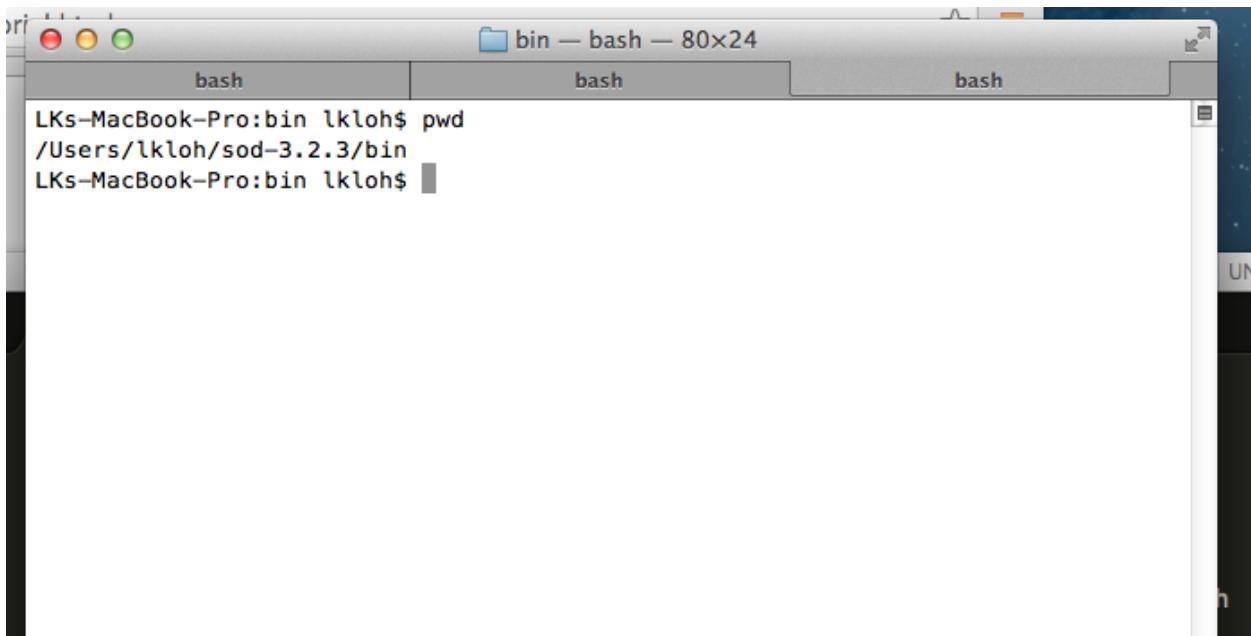
First, download SOD.

Once you have gotten the folder for SOD, put it somewhere where you won't touch it too much. What I did was put the SOD folder in my home directory, though other places are acceptable as well, as long as it is not too easy to delete it by accident.



```
LKs-MacBook-Pro:sod lkloh$ cd
LKs-MacBook-Pro:~ lkloh$ ls
AI-Intro          Documents          Pictures
Cascades-2014     Downloads         Public
Cascades-BigFiles Dropbox          classify
Cascades-Fall-2013 Library          pysmo
Cascades-SESUR-2012 Movies           sod-3.2.3
Desktop           Music
LKs-MacBook-Pro:~ lkloh$
```

Once you have it there, get the path to the sod folder's bin and put it in your path folder.



```
LKs-MacBook-Pro:bin lkloh$ pwd
/Users/lkloh/sod-3.2.3/bin
LKs-MacBook-Pro:bin lkloh$
```

Inside my home directory's bash profile (you get there by typing `cd`), you put the path to `sod-3.2.3/bin` by adding it to either the `bashrc`, `bash_profile`, or `profile` files.

5.2.2 Example SOD recipe

Inside the repository `data-example`, there is a folder `sod_requests`. The file within it called `sod_request.xml`, which is available [here](#), is an example of a sod request recipe that will download data from IRIS. To run it, cd into the folder containing `sod_request.xml`, and do:

```
sod sod_request.xml
```

Downloading the data (output as SAC files) may take a while. This receipt filters the data, and outputs the folders `processedSeismograms` and `seismograms`, which contain the filtered and unfiltered data.

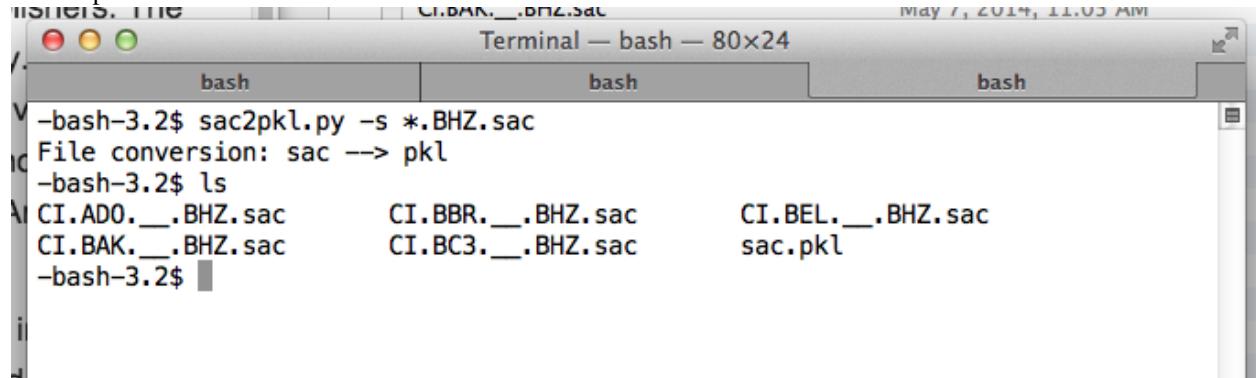
SAC INPUT/OUTPUT PROCEDURES FOR AIMBAT

6.1 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:

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

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



A screenshot of a Mac OS X terminal window titled "Terminal — bash — 80x24". The window has three tabs, all labeled "bash". The terminal shows the command `sac2pkl.py -s *.BHZ.sac` being run, followed by the message "File conversion: sac --> pkl". Then it lists the input files: CI.ADO.___.BHZ.sac, CI.BAK.___.BHZ.sac, CI.BBR.___.BHZ.sac, CI.BC3.___.BHZ.sac, and CI.BEL.___.BHZ.sac. Finally, it shows the output file `sac.pkl`.

6.2 Converting from PKL to SAC files

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

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

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

The screenshot shows a terminal window with the title bar 'example_pk1_files — bash — 80x24'. The window contains the following text:

```
dhcp-10-101-89-46:example_pk1_files lkloh$ ls
20110915.19310408.bhz.pkl      20120101.05275598.mcp
20120101.05275598.bhz.pkl      loc.sta
dhcp-10-101-89-46:example_pk1_files lkloh$ sac2pk1.py --p2s 20110915.19310408.bhz.pkl
File conversion: pk1 --> sac
dhcp-10-101-89-46:example_pk1_files lkloh$
```

FILTERING DATA

There are several options for filtering data. SAC offers several ways to filter data that are not discussed here. You could also do so when downloading data using SOD. Alternatively, you could filter data in AIMBAT.

7.1 Filtering GUI

To filter data, once you have imported the files to run into the GUI, hit the `filter` button to bring up the filtering GUI. The figure on top shows the results of the signal plotted against time before and after filtering with a butterworth filter.

You can adjust the corner frequencies used on the figure on the lower corner. The defaults used here 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. By clicking on the lower figure, you can select the low frequency and the high frequency you want. Click `apply` to filter the seismograms when you are satisfied with the filter paramters chosen.

7.2 Saving Options

There are several options for saving the SAC header files of the seismograms you have chosen.

**CHAPTER
EIGHT**

ANALYZING DATA

8.1 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.

PARAMETER CONFIGURATION

9.1 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 \(Application Programming Interface\)](#)

AIMBAT uses the default toolkit Tk and backend TkAgg.

Visit these pages for an explanation of the [backend](#) and how to customize it.

9.2 Configuration File

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

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.

9.2.1 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 <i>plotphase.py</i> |
| 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 = xcorr90 | Module for calculating cross-correlation: xcorr for Numpy or xcorr90 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 |
| filename = mc | Output file name of MCCC. |
| xcorr_modu = xcorr90 | Module for calculating cross-correlation: xcorr for Numpy or xcorr90 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 = lnc0 | Type of lsqr solution: weighted by correlation coefficient, solved by lapack |
| #lsqr = lnre | Type of lsqr solution: weighted by residual, solved by lapack |
| rcfile = .mccrc | 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 |

SAC DATA ACCESS

10.1 Python Object for SAC File

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 their attributes. Modifications of object attributes are saved to file. It is written purely in Python so that it also runs with `Jython`.

10.1.1 `egsac.py`

The `<pkg-install-dir>/aimbat/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/example_pkl_files/Event_2011.09.15.19.31.04.080`, hereafter referred to as `<example-event-dir>`.

```

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()

```

10.1.2 Resampling Seismograms

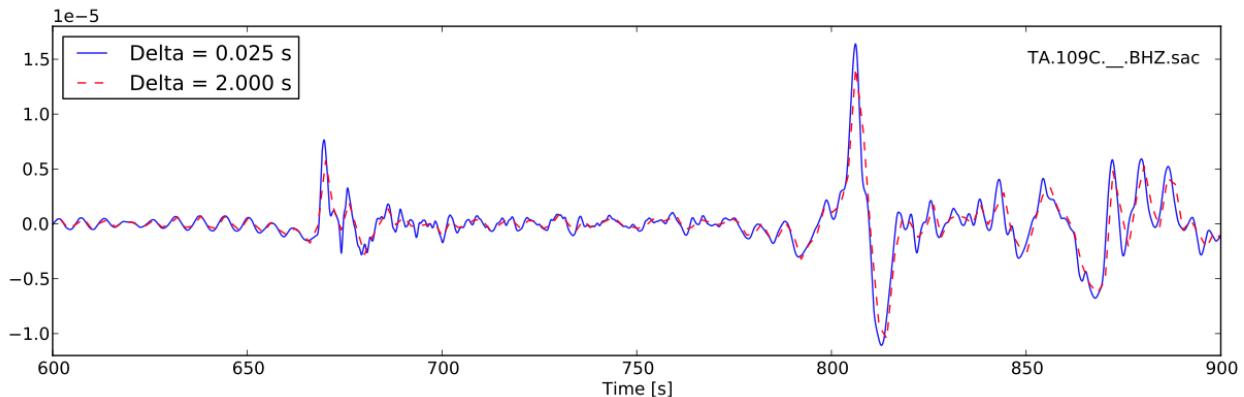
In this example, a SAC file named TA.109C._.BHZ.sac is read in as a sacfile object. The time array is calculated from SAC headers. The data array is resampled from interval 0.025 to 2.0 seconds using Scipy's signalprocessing module.

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

```

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

```



10.2 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 serve this purpose by reading a SAC file and returning a `sacd` object that is very similar to the `sacfile` object. Essentially, the `sacd` 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 `sacd` 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 at command line:

```
sac2pkl.py -h
```

and the result is displayed in the figure below. 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.

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

```
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.

10.3 SAC Plotting and Phase Picking

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

Options:

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

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

Options:

| | |
|--|--|
| <code>-h, --help</code> | show this help message and exit |
| <code>-f FILL, --fill=FILL</code> | Fill/shade seismogram with positive (1) or negative (-1) signal. Default is none (0). |
| <code>-r RELTIME, --relative-time=RELTIME</code> | Relative time to a time pick header (t_0-t_9). Default is -1, None, use absolute time. |
| <code>-u, --upylim</code> | Update ylim every time of zooming in. |
| <code>-k, --pick</code> | Plot time picks. |
| <code>-w, --twin</code> | Plot time window. |
| <code>-x XLIMIT, --xlimit=XLIMIT</code> | Left and right x-axis limit to plot. |
| <code>-y YNORM, --ynorm=YNORM</code> | Normalize ydata of seismograms. Effective only for positive number. Default is 2.000000. |
| <code>-Y, --ynormtwin</code> | Normalize seismogram within time window. |
| <code>-S SRATE, --srate=SRATE</code> | Sampling rate to load SAC data. Default is None, use the original rate of first file. |
| <code>-b, --boundlines</code> | Plot bounding lines to separate seismograms. |
| <code>-n, --netsta</code> | Label seismogram by net.sta code instead of SAC file name. |
| <code>-m MAXNUM, --maxnum=MAXNUM</code> | Maximum number of selected and deleted seismograms to plot. Defaults: 25 and 5. |
| <code>-s SORTBY, --sortby=SORTBY</code> | 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. |

```
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()
```

10.3.1 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, sacp1.py  
sacplot.py -z, sacp2.py  
sacplot.py -d, sacprs.py  
sacplot.py -a, sacpaz.py  
sacplot.py -b, sacpbaz.py
```

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:

```
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
```

or:

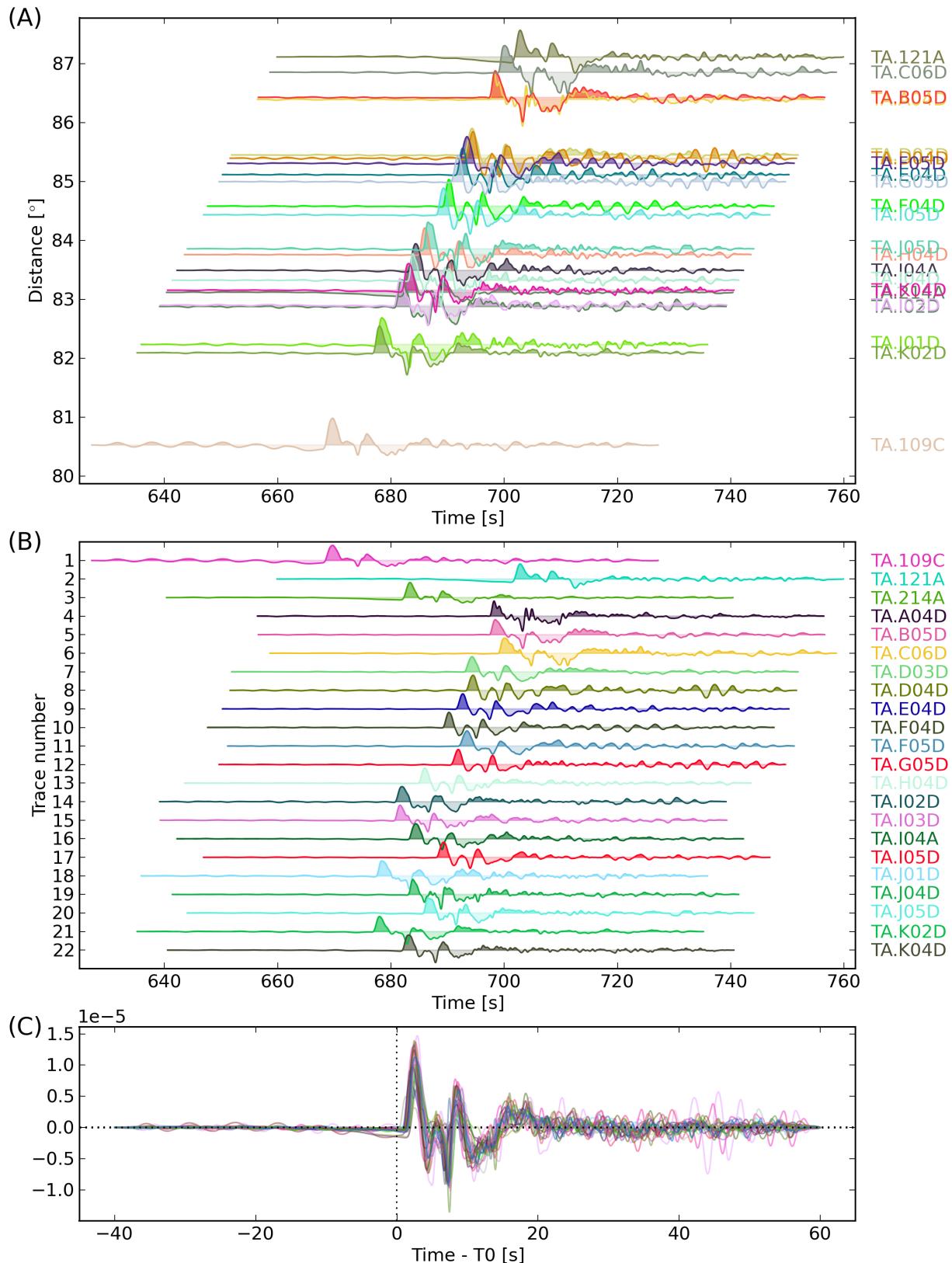
```
sacp1.py bhz.pkl
```

For large numbers 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 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.relttime=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.

10.4 SAC Phase Picking

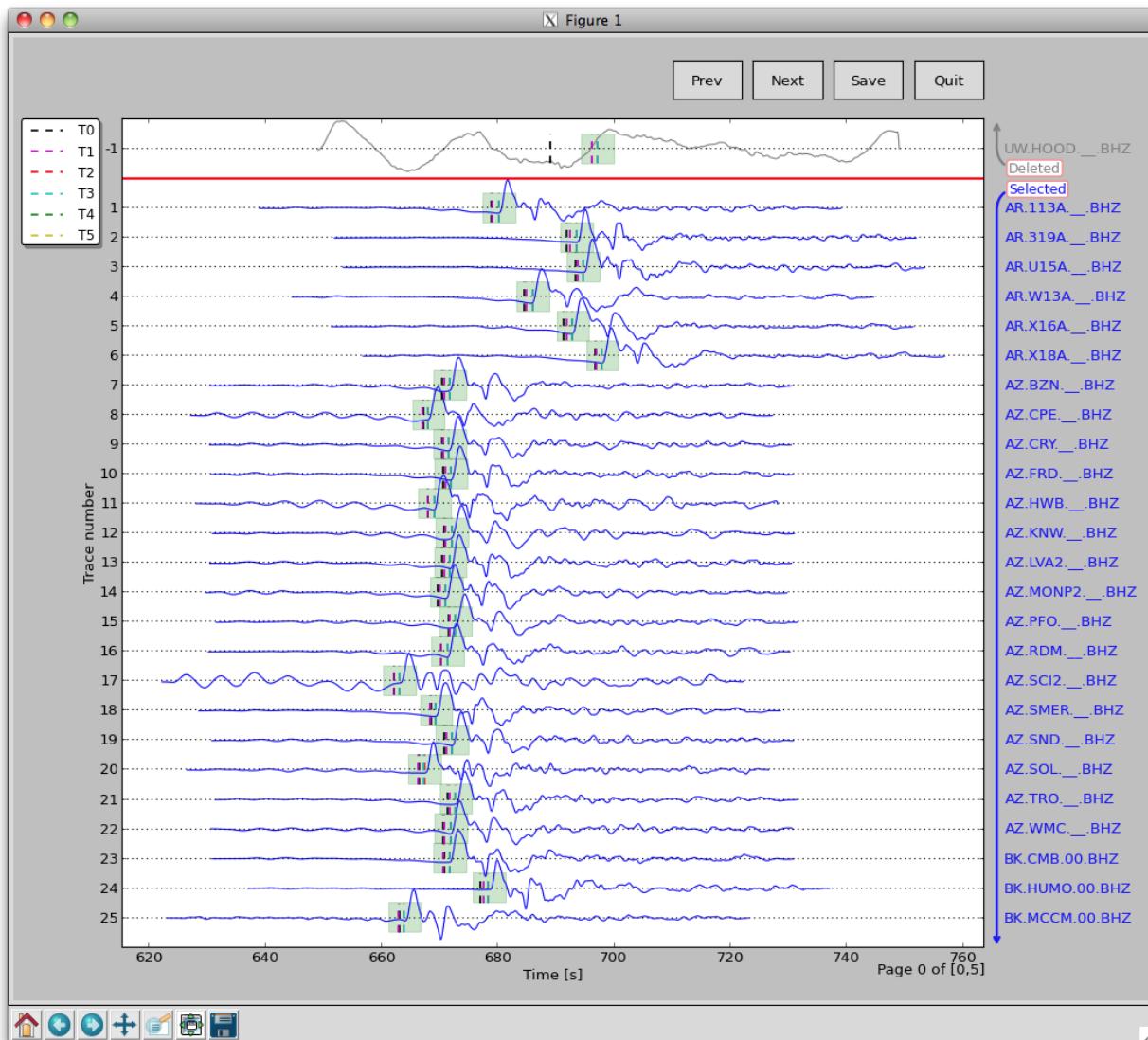
SAC plotting (`pysmo.aimbat.plotphase`) does not involve change in data files, but phase picking (`pysmo.aimbat.pickphase`) does. A GUI is built for the user to interactively pick phase arrival times. The figure below is an example screen shot running:

```
sacppk.py 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 `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 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:

```
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
```

The help message of the iccs.py 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 MINSEL, --minsel=MINSEL | Minimum number of selected seismograms for auto selection. Default is 5. |

The help message of the mccs.py 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, --filename=OFILENAME Output file name. Default is \$evdate.mc\$phase
- a, --allseis Use all seismograms. Default to use selected ones.

CHAPTER
ELEVEN

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.

11.1 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 $_3T_i$.

1. Coarse alignment by ICCS
2. Pick phase arrival at the array stack
3. Refined alignment by ICCS
4. 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 11.1: 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 |
| 1. | ICCS | W_a | $_0T_i$ | T0 | $_1T_i$ | T1 |
| 2. | ICCS | W_b | $_2T'_i$ | T2 | $_2T_i$ | T2 |
| 4. | 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.

11.2 Picking Travel Times

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

11.2.1 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.

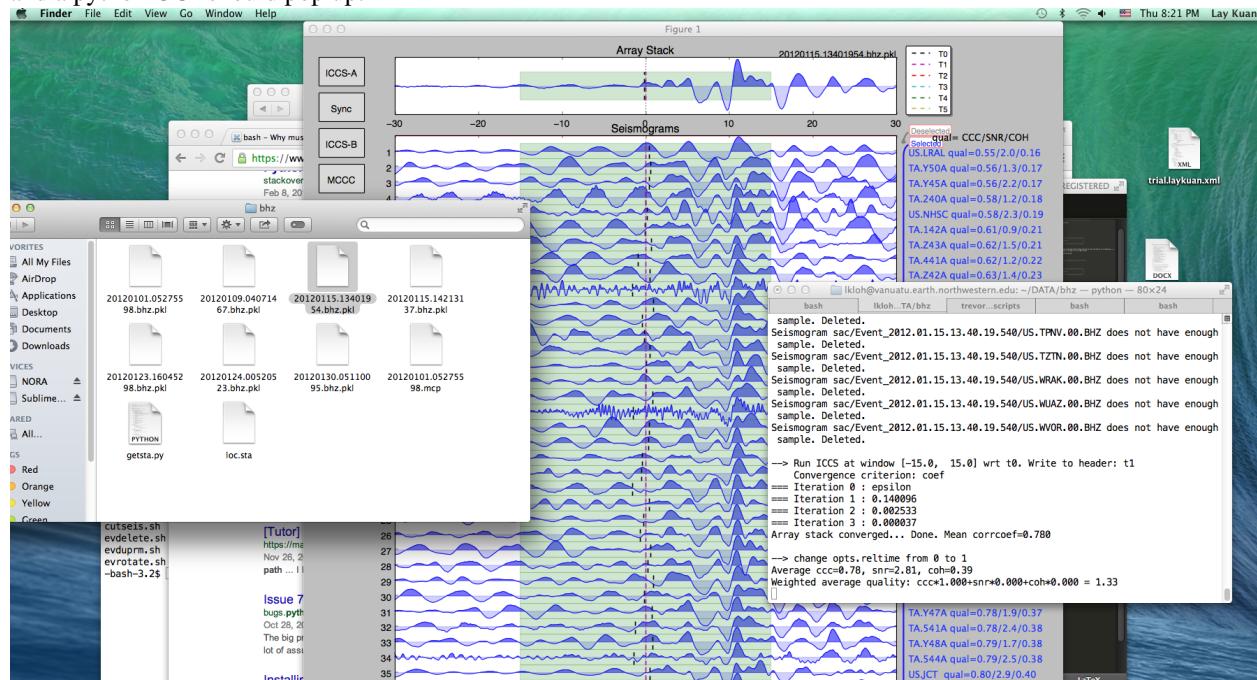
11.2.2 Running `ttpick.py`

Run `ttpick/py <path-to-pkl-file>`. 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 the 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:

```
ttpick.py 20110915.19310408.bhz.pkl
```

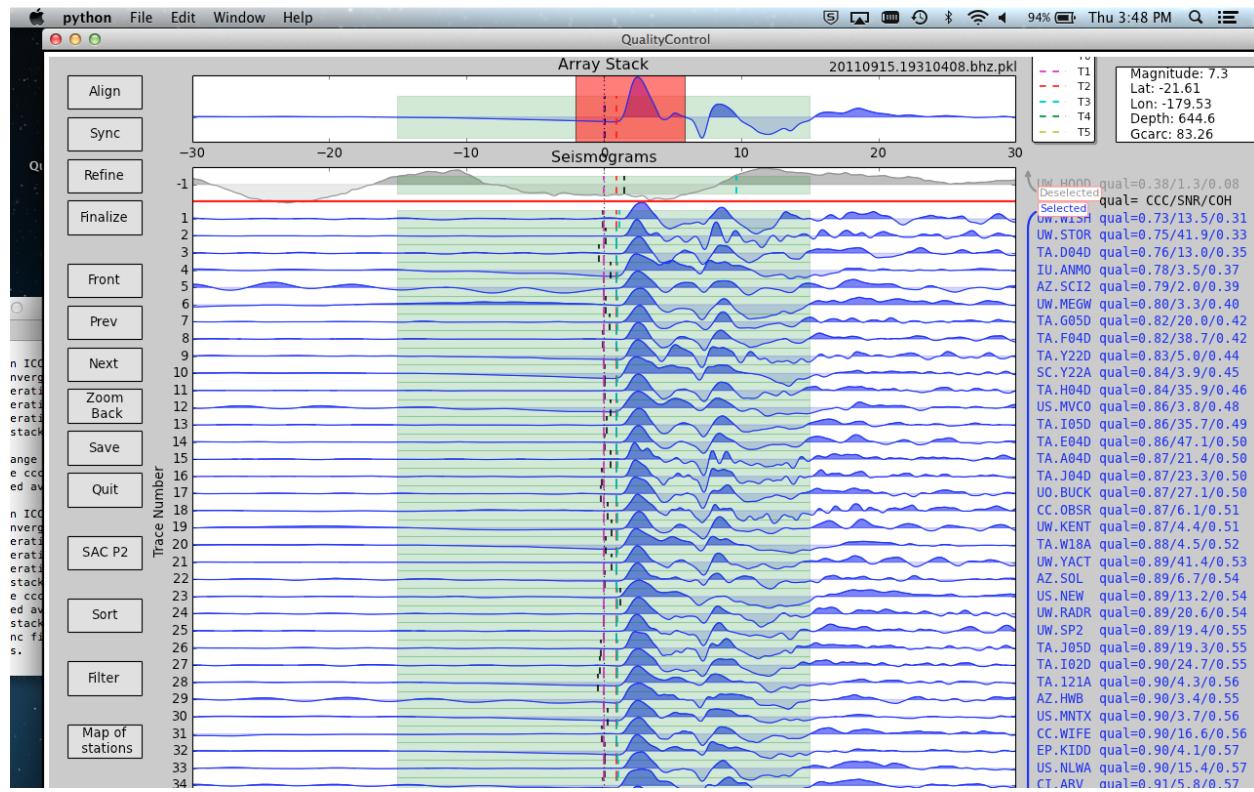
and a python GUI should pop up.



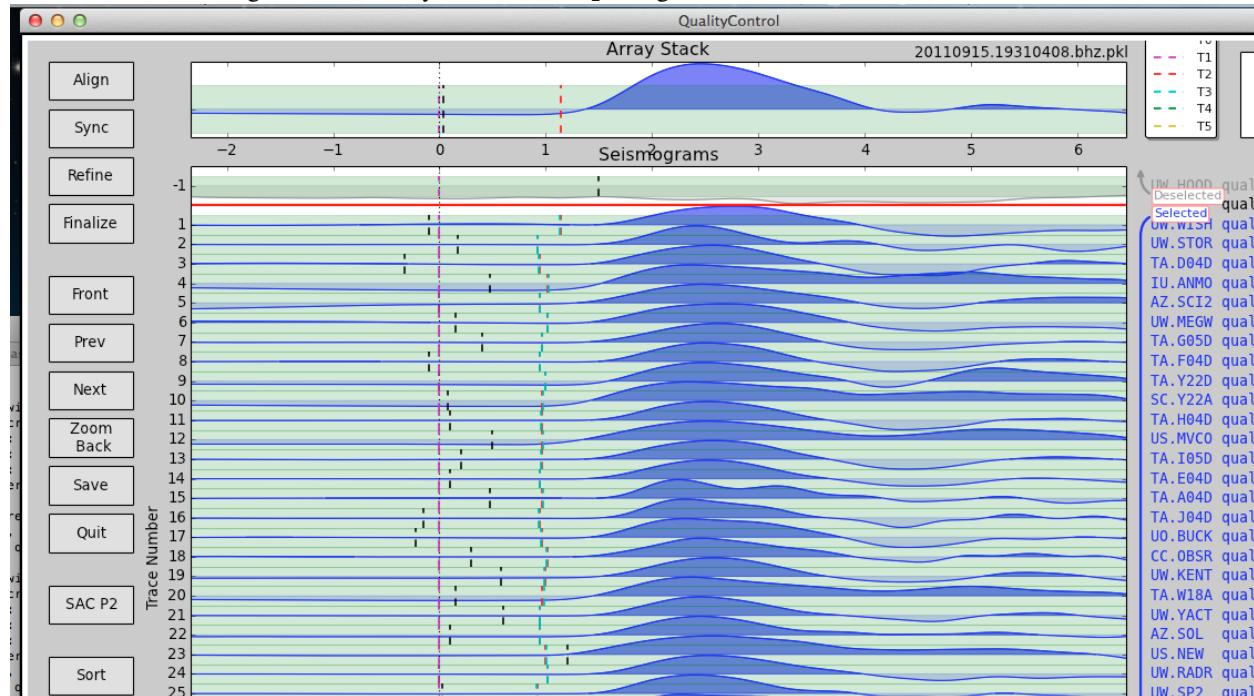
11.2.3 Align

Align is only used in the beginning, if you have altered some of the travel time arrivals of the seismograms by pressing `t2` and want to realign the array stack.

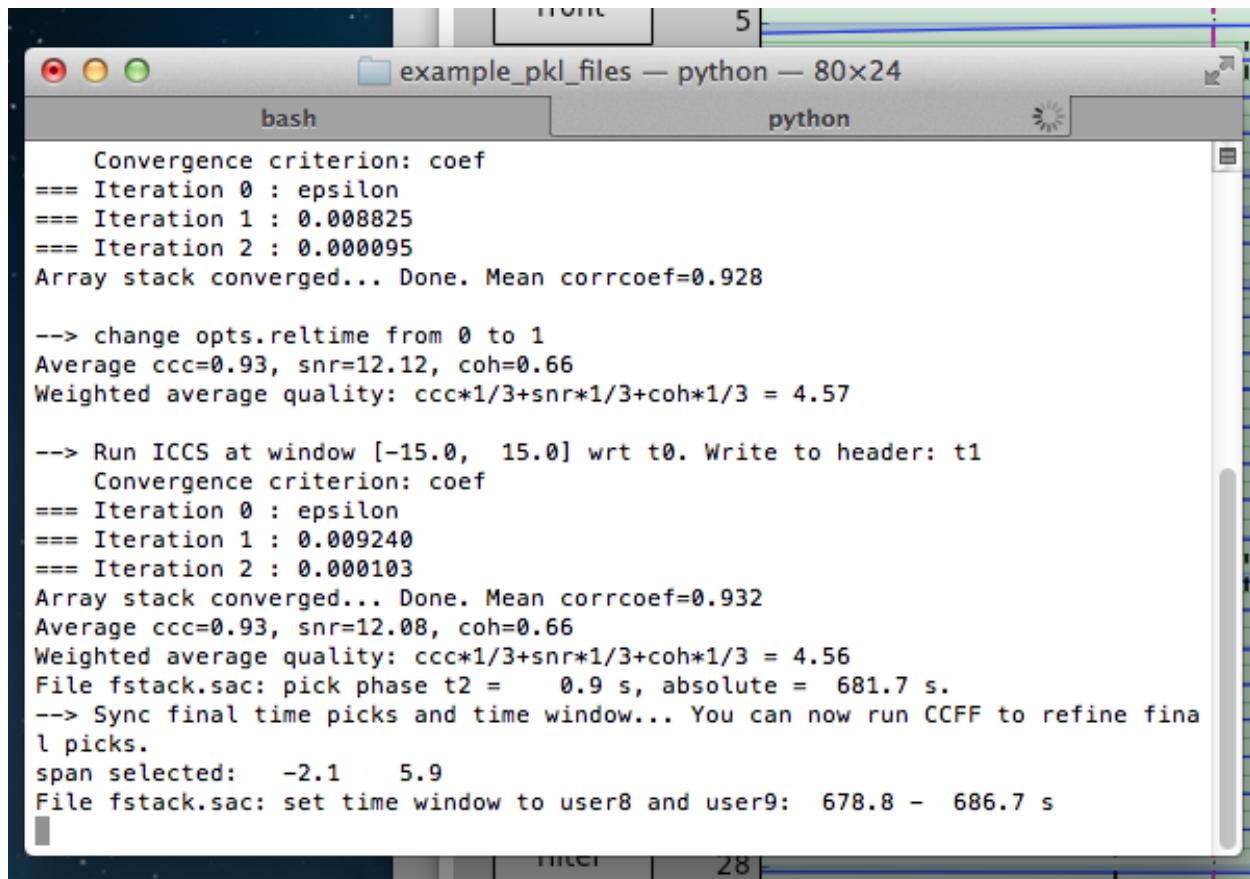
11.2.4 Selecting a time window around the arrival of interest



Hit the Align button and use t_2 to select the arrival time. Now press Sync. Use the mouse to select the desired time window on the seismogram on the array stack. Press Sync again.



Next, set the mouse over the seismogram and press the w key. If the new time window has been saved, a message noting the new size of the time window should be printed in the terminal.



```

Convergence criterion: coef
== Iteration 0 : epsilon
== Iteration 1 : 0.008825
== Iteration 2 : 0.000095
Array stack converged... Done. Mean corrcoef=0.928

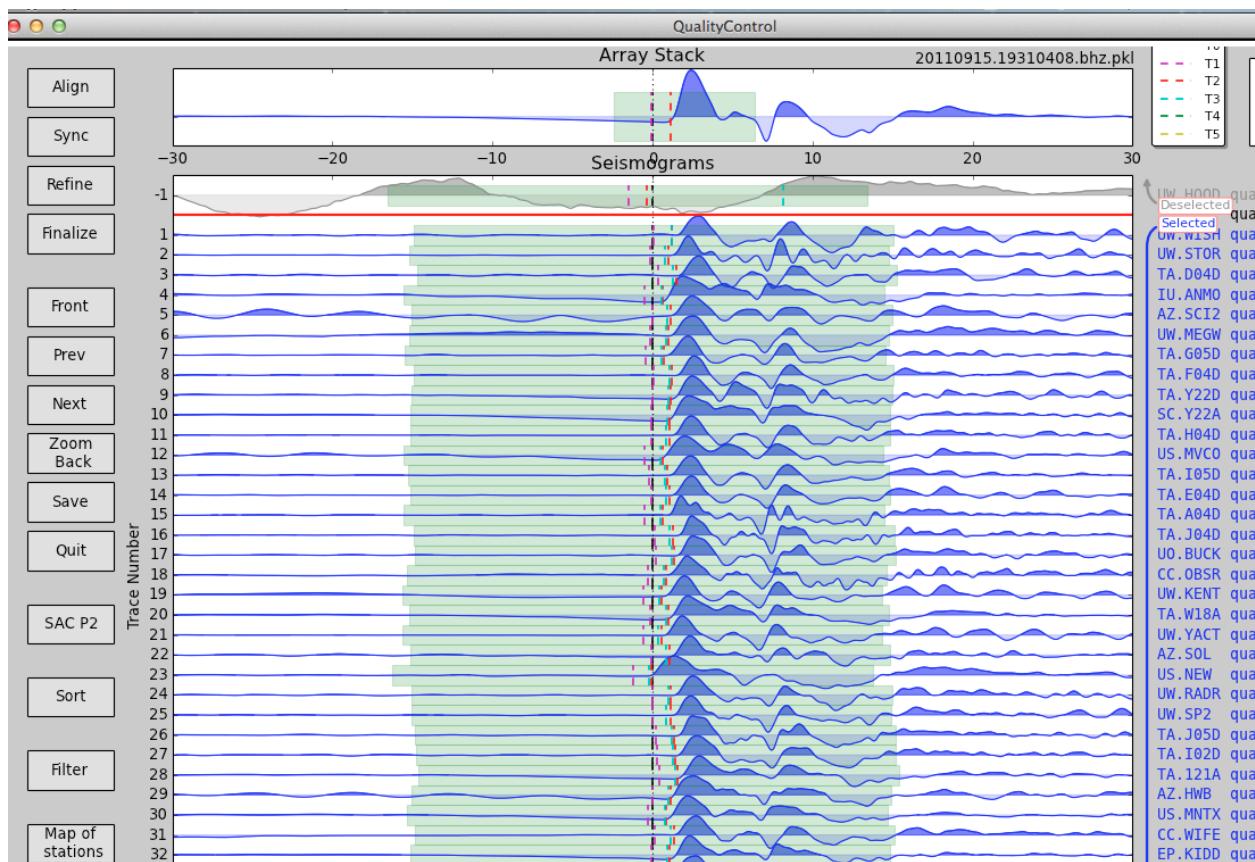
--> change opts.reltim from 0 to 1
Average ccc=0.93, snr=12.12, coh=0.66
Weighted average quality: ccc*1/3+snr*1/3+coh*1/3 = 4.57

--> Run ICCS at window [-15.0, 15.0] wrt t0. Write to header: t1
    Convergence criterion: coef
== Iteration 0 : epsilon
== Iteration 1 : 0.009240
== Iteration 2 : 0.000103
Array stack converged... Done. Mean corrcoef=0.932
Average ccc=0.93, snr=12.08, coh=0.66
Weighted average quality: ccc*1/3+snr*1/3+coh*1/3 = 4.56
File fstack.sac: pick phase t2 = 0.9 s, absolute = 681.7 s.
--> Sync final time picks and time window... You can now run CCFF to refine final picks.
span selected: -2.1 5.9
File fstack.sac: set time window to user8 and user9: 678.8 - 686.7 s

```

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. Click the Save Headers Only button.

Quit the GUI and restart it, and you will see that your new, changed time window is preserved in green in the array stack.



Now press refine and all the seismograms will align with the smaller time window.



11.2.5 Get rid of bad seismograms

If there are any really bad seismograms, you can click on them to deselect them. Bad seismograms are those that look nothing like the shape of the array stack pictured. Usually there are more than enough seismograms, so it is safe to throw out any that deviate more than a bit from the array stack.

11.2.6 Filtering

To filter your data, hit the `filter` button, and a window will popup for you to use the [Butterworth filter](#) to filter your data.

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

You can choose the order by selecting one of the values provided (default is 1), and choose the low and high frequencies for bandpassing by clicking on the appropriate start and stop frequency on the lower graph.

11.2.7 Refine

Hit the `Refine` button to begin the initial cross-correlations. These appear as red lines.

We are not using `Align` here, but these are the theoretical arrival times, marked in black.

11.2.8 Finalize

Hit `Finalize` to run the Multi-Channel cross-correlation. Do not hit `Align` or `Refine` again, or all your work will be erased. A warning will pop up to check if you really do want to hit these two buttons if you do click on them.

11.2.9 Manually pick the arrival times using t2

For an earthquake, it is expected that the arrival times should be identical in an idealized situation. However, since stations are located in 3D space, this is not necessarily the case. For earthquakes of magnitude 7.0 and above, usually the arrival times are very well aligned as the signal is high. However, if the earthquake is too strong, the source gets complicated, so it needs filtering.

Below a magnitude of 6.0, the signal to noise ratio gets very weak. If the weighted average quality gets too low (1.0 and below), it may not be worth keeping that data set unless you really need it.

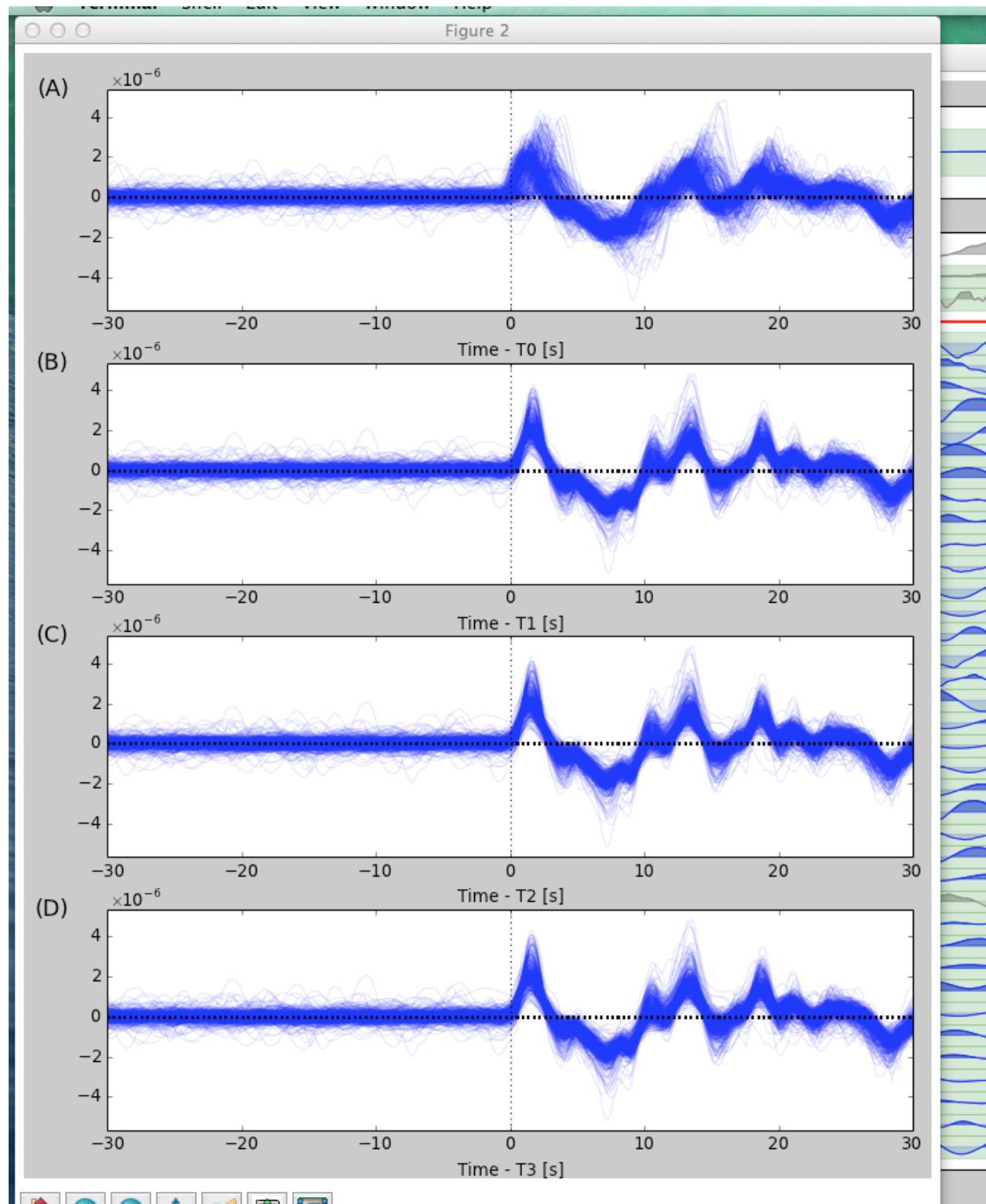
We manually pick the arrival times to align them. Click on the GUI window, hover over the correct spot where you want to pick the new travel time, and type `t2`. A red line should appear exactly where your mouse was. You can zoom in to help you with this picking.

Also pick the arrival time on the array stack. For the arrival times, you want to align the point where the first peak occurs most of all, then try to get the peaks to align.

11.2.10 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.

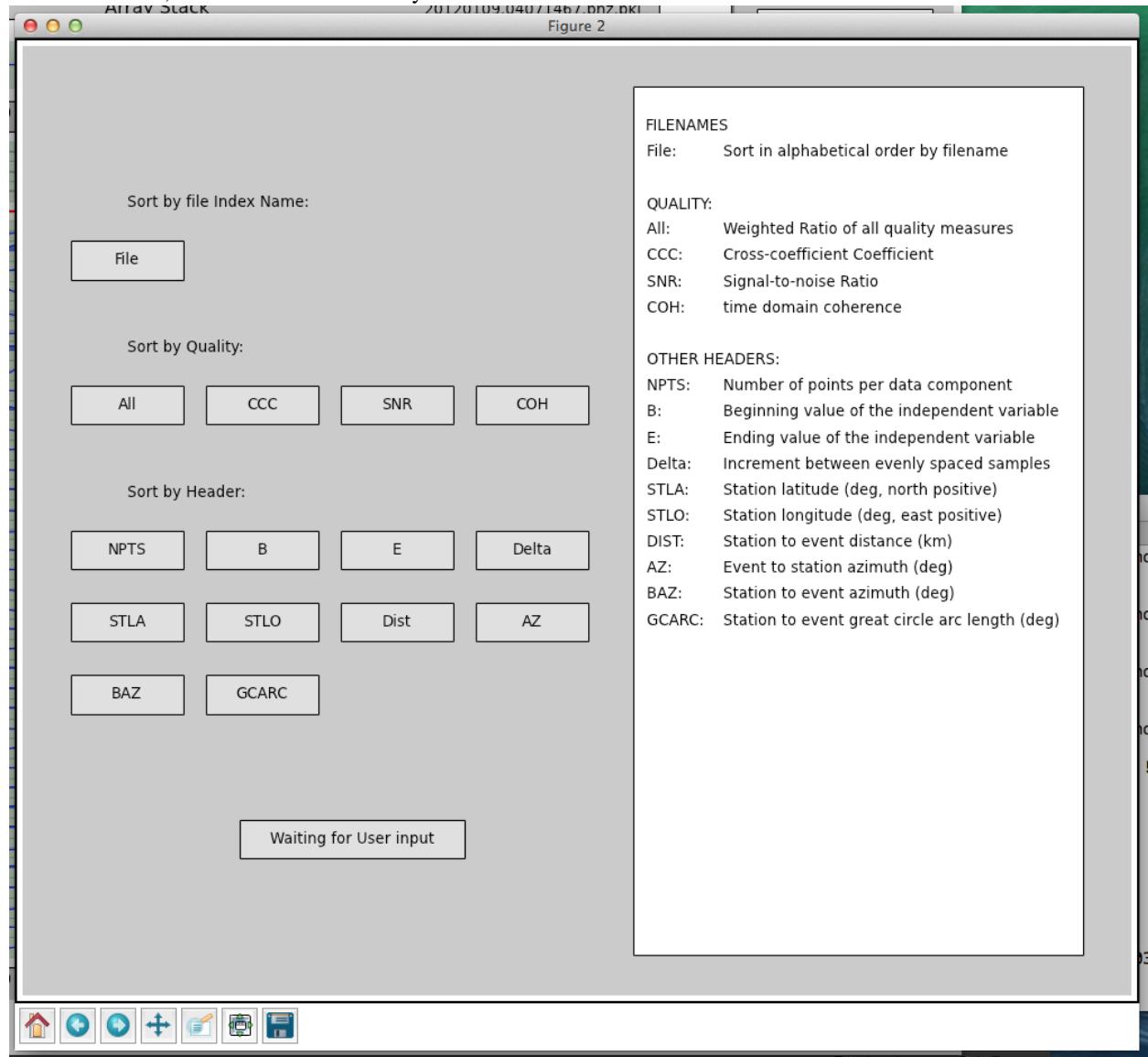
Click on the outliers that stray from the main group of stacked seismograms. The terminal will output the names of the seismograms that you clicked on, so you can return to the main GUI window and readjust the travel times. Note: hitting `SACP2` before hitting `finalize` will often cause AIMBAT to close,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



11.2.11 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 so that you can find the bad seismograms 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 to sort the files by filename.



The seismograms are stretched to fit together, but they may be scaled differently.

11.3 What the Alignments Stand For

- T0: Theoretical Arrival

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

11.4 Post Processing

11.4.1 Getting the output

In the same folder as the initial PKL file you ran `ttpick.py` 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 $t3 + \text{average arrival times}$, and `t0_times` are the theoretical arrival times. `delay_times` are obtained by taking $t3 - t0$.

11.4.2 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.

11.4.3 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 `pk1` files chosen after you saved to. To do this, type `./getsta.py *.pk1`.

```

-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

```

bugs.python.org/issue/

49 Run \ve

11.4.4 Picking Travel Times does not work

If you run `ttick.py <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
```

Comment out the other backends!

11.4.5 Travel Times

If one of the seismograms being picked does not fit completely within the green (computer) window, and you hit *ICCC-A* or *ICCC-B*, you will get an error message complaining about the exact seismogram which is too short. Simply deselect it.

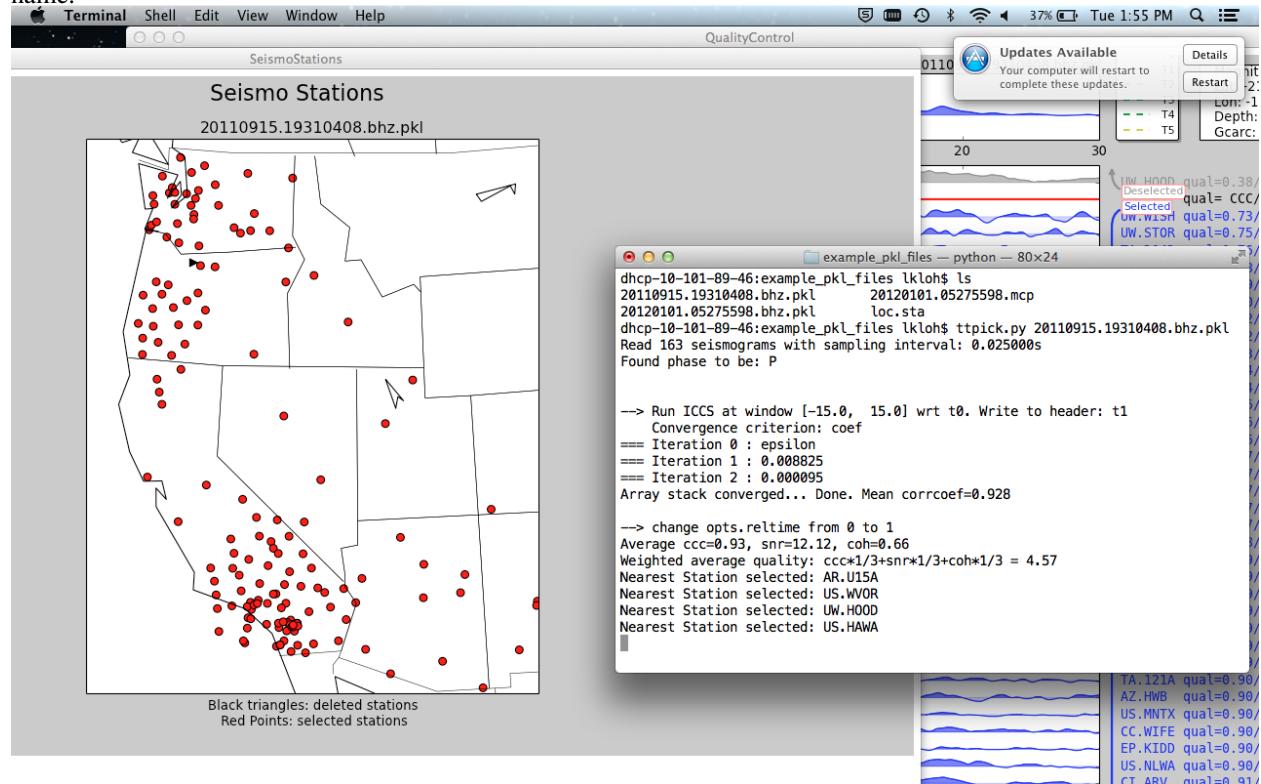
CHAPTER TWELVE

VISUALIZING STATIONS ON A MAP

After running:

```
ttpick.py <sac-files>
```

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



CHAPTER
THIRTEEN

UNIT TESTING

This section is mainly for those who wish to make tweaks to AIMBAT themselves. We have added some unit tests to AIMBAT to ensure that it is robust. See the [Python Unit Testing Framework](#) for more details.

13.1 Running the Tests

In the AIMBAT repository, cd into `/src/pysmo/unit_tests` and run:

```
python run_unit_tests.py
```

CHAPTER
FOURTEEN

UPDATING THIS MANUAL

This is for someone who wants to be a collaborator on AIMBAT only. This is NOT necessary for anyone who only wants to use AIMBAT. AIMBAT will work fine if you do not install the dependencies listed here.

To be able to update the manual, download the [source code](#) from Github, and install the dependencies.

14.1 Dependencies

- *Sphinx*. Download and install from [here](#). Don't get the Python Wheel version unless you know what you are doing
- *LaTeX*. Download it from [here](#). Get the package installer.
- A browser. But if you are reading this, you already have it.

14.2 How to update this manual

On the master branch, cd into the github repository *aimbat-docs* <<https://github.com/pysmo/aimbat-docs>> and run:

```
sphinx-build -b html . builddir  
make html  
make latexpdf
```

The first two commands build the html for the webpage, while the last command makes a pdf version of the online documentation.

Now, commit the changes made in github, and push the changes to the master branch. The changes should be visible in the documentation within a few minutes.

**CHAPTER
FIFTEEN**

CITATIONS

CHAPTER
SIXTEEN

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.
- [LouVanDerLee2013] 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.