AIMBAT Documentation

Release 0.1.2

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

June 16, 2014

Contents

Contents:

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: lkloh@cmu.edu

- Xiaoting Lou

  Email: xlou@u.northwestern.edu

- Suzan van der Lee

  Email: suzan@earth.northwestern.edu

TWO

INSTALLING PYTHON DEPENDENCIES

Usually, Macs already have python installed by default. To check if you have python on your mac, open up terminal, and do python in the terminal. If python is installed you should see a python console show up, displaying the version number of python. The version number should be at least 2.7 or higher. If python is not installed, you should see an error message show up. However, you may or may not have some necessary packages installed.

2.1 Getting Git

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

The best way to get Git is to use Git for Mac. This is a GUI you can used for your repositories, but if you want to use the command line interface you can also enable it when setting it up.

2.2 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 construct 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.
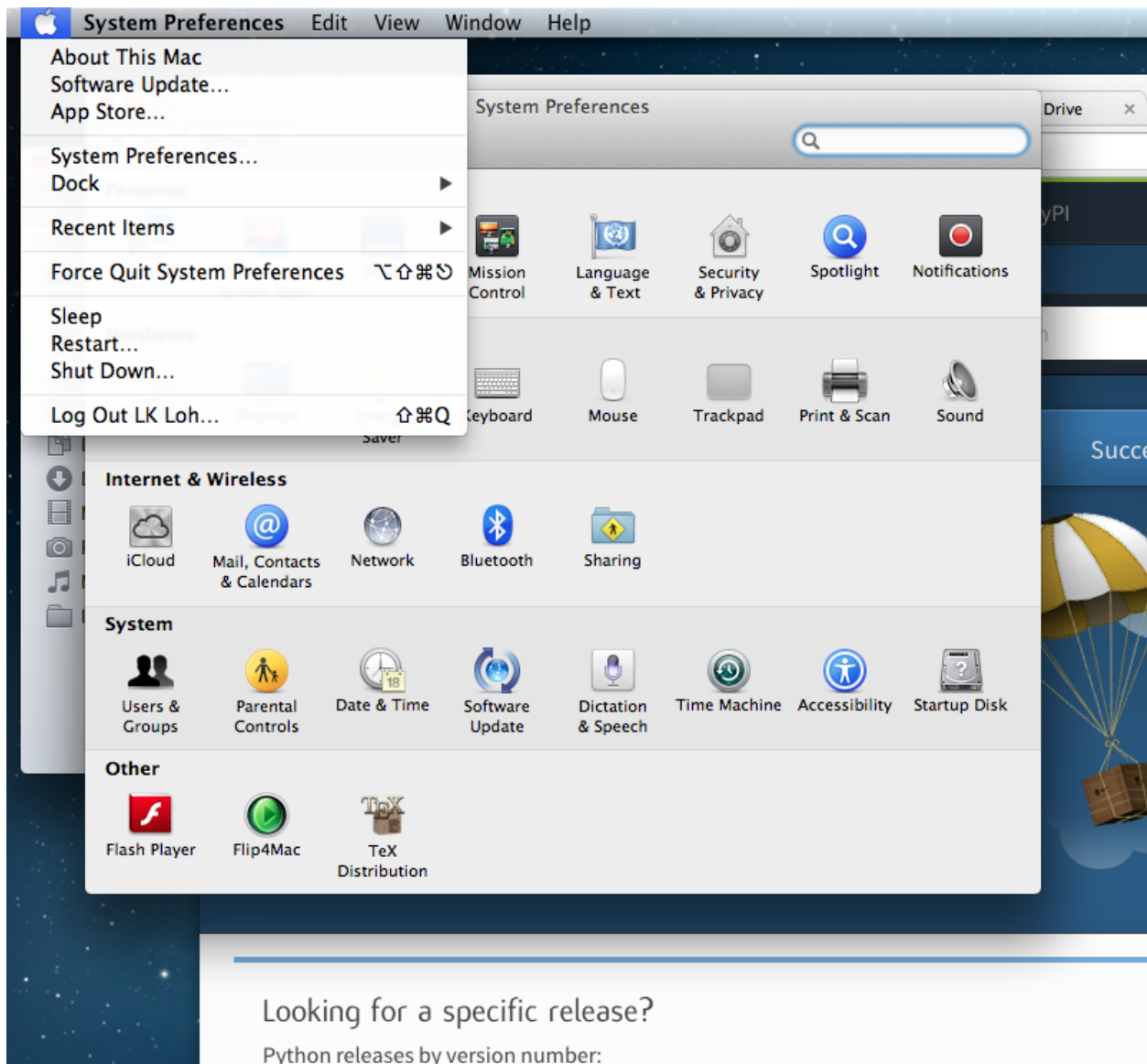
2.3 Python Packages needed

The following are required for AIMBAT * Numpy * Scipy * Matplotlib

Using iPython is optional but recommended, as iPython is an interactive console designed to make

2.4 Installing Python with Enthought Canopy

Shaowei Lin suggested Enthought Canopy to install all the Python packages easily. If you download the free version of Express Enthought Canopy, it gives you everything you need for installing AIMBAT properly.

If you do not want to use Enthought Canopy, read the rest of this section.

2.5 Installing Python with Macports

Usually, Macs already have python installed by default. To check if you have python on your mac, open up terminal, and type `python`. If python is installed, a console should pop up. If python is not installed, you should see an error message show up. You should download the correct verion for your operating system here.

2.5.1 Getting Macports

Its best to use Macports to install the necessary python libraries for AIMBAT. If you just upgraded your operating system, you need to upgrade Macports and re-install the libraries as well.

2.5.2 Installing the necessary components

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

```
sudo port install python27
sudo port install py27-numpy
sudo port install py27-scipy
sudo port install py27-matplotlib
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
```

You need this version, not other versions on your computer, since this is the one that has the libraries AIMBAT needs.

2.6 Regarding Package Managers

The package manager brew caused many problems when tried. If you figured it out properly, please *contact the authors* with instructions~ In general, the authors do not recommend trying to install the packages separately when there are Python versions that will come with all the packages pre-installed already.

Scipy is especially tricky as it relies on Fortran and C as well. The authors of scipy recommend using Enthought Canopy or Anacoda to install it.

2.7 Installing Basemap

Disclaimer: Lifted from content written by this guy with some tweaks.

Enthough Python should get you most of the dependencies needed. 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. Export the directory where `Geos` was installed:

```
export GEOS_DIR=/usr/local/Cellar/geos/3.4.2/
```

And install Basemap:

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

To check it worked, at the terminal, do:

```
python
```

and then:

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

2.8 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*.

2.8.1 Macport

You may 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.

2.8.2 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.

2.8.3 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 and doing `ls -a` to see all the hidden files, and open then by doing `vi .bashrc` in vim, for instance. 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.

2.8.4 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.
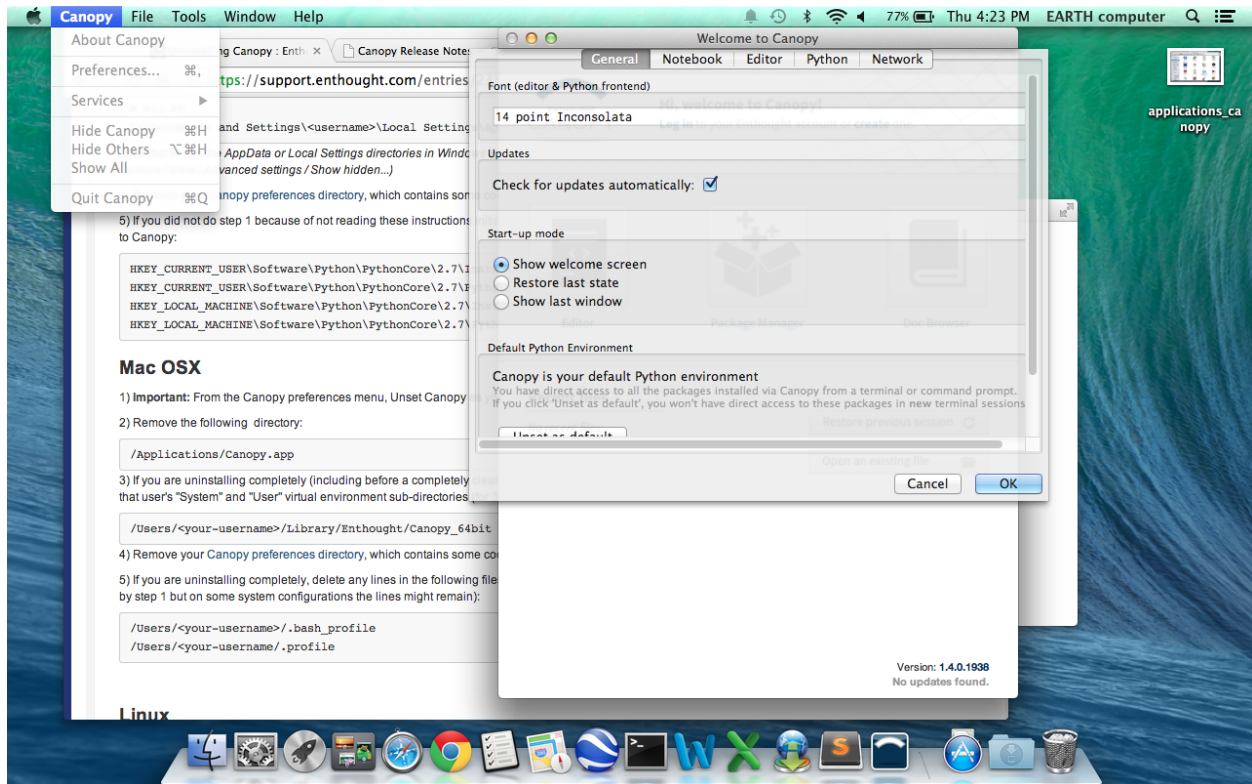
2.8.5 Installing Enthought Canopy

Occasionally, Enthought Canopy may not open the default setup environment after you downloaded and tried to install it. If this happens, open the Canopy package, go to "Preferences", and select Canopy as your default environment.
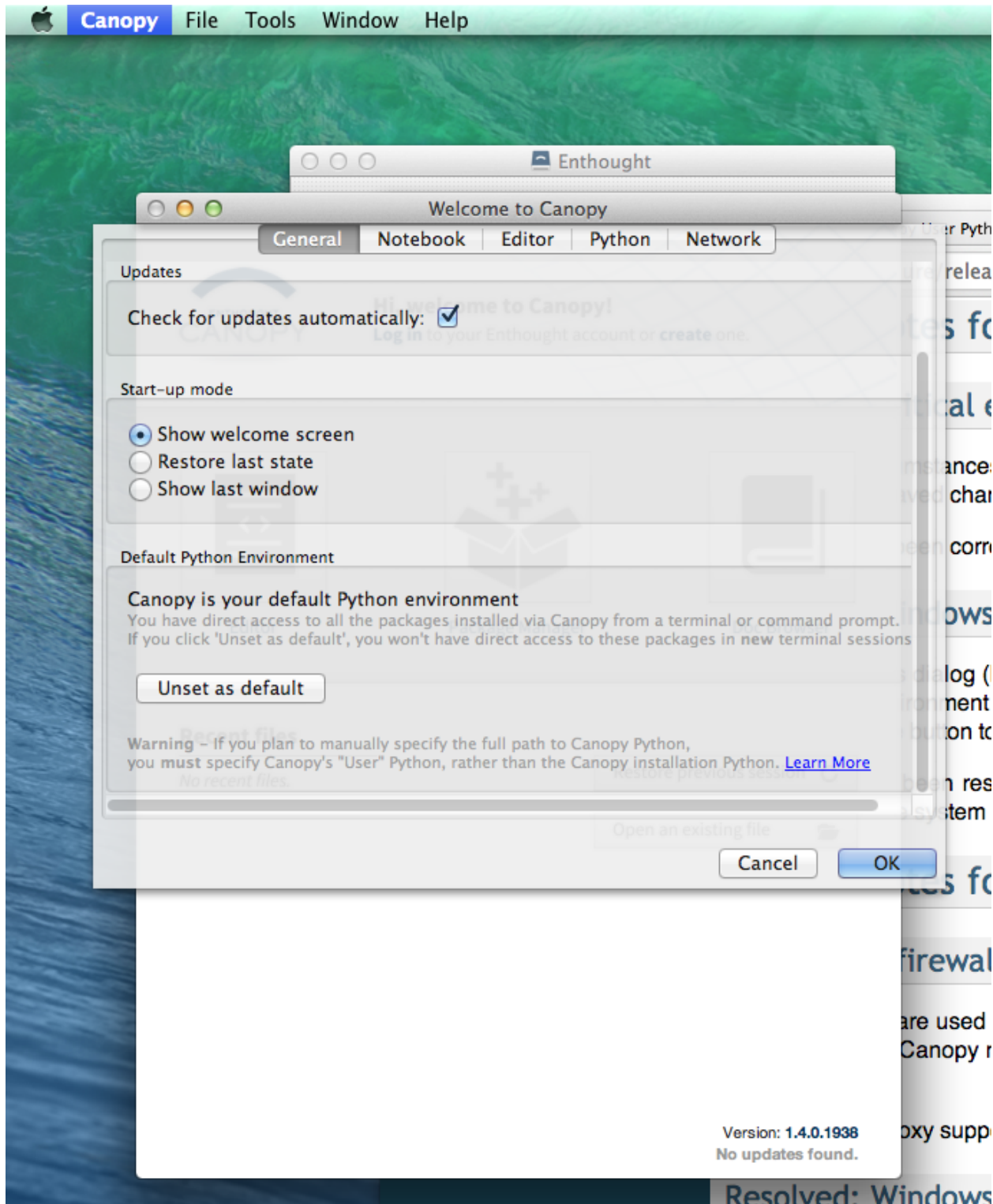
### 2.8.6 Uninstalling Enthought Canopy

The official Enthought gives suggestions on uninstalling here.

STEPS:

1. From the Canopy preferences menu, unset Canopy as your default Python.

2. For each Canopy user, delete the following directory which contains that user's "System" and "User" virtual environment subdirections.

3. Delete Canopy from the Applications folder.

4. Clean up the hidden files. Delete anything referencing Canopy or Enthought in the hidden files, as evidence by referencing `ls -a` in your home directory. Check the `.bashrc` and `.profile` directories first. If Enthought is not completely gone, this happens if you call Python.

5. (Optional). Keep doing `which python` and cleaning the python files that show up, until `which python` gives you nothing when you type it in the terminal.
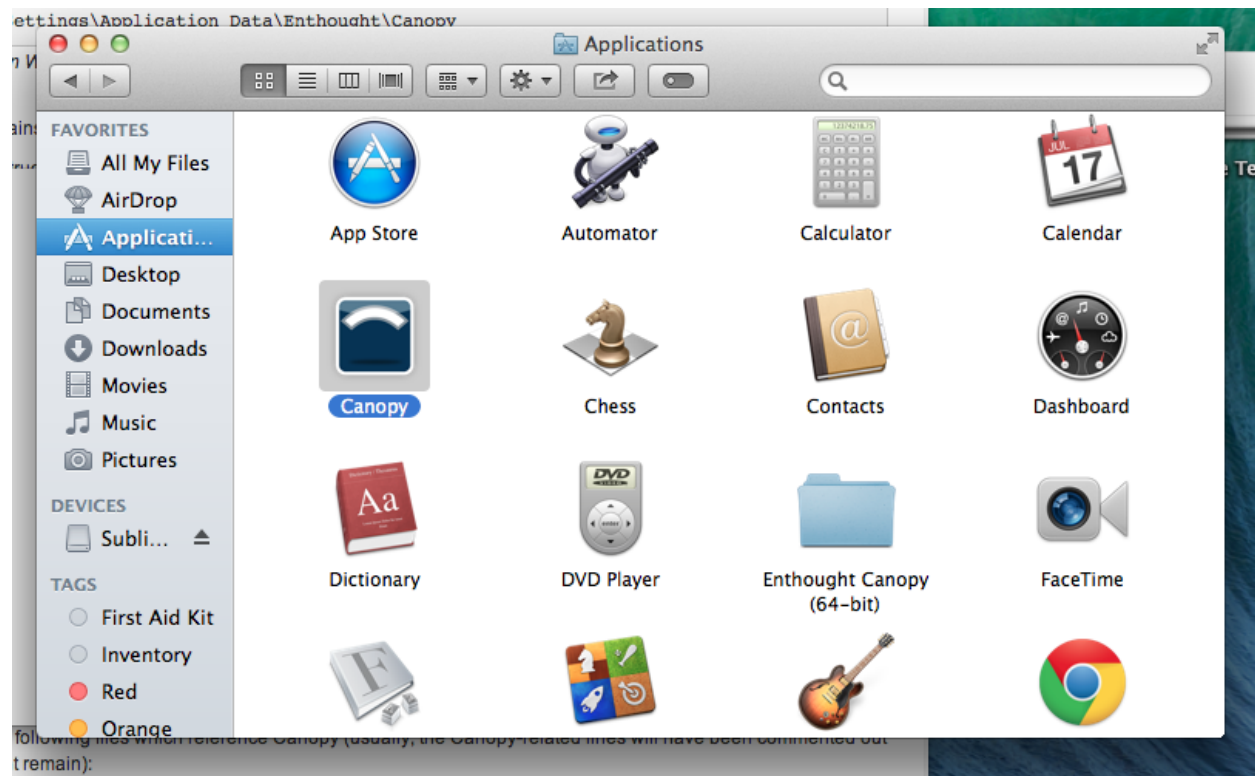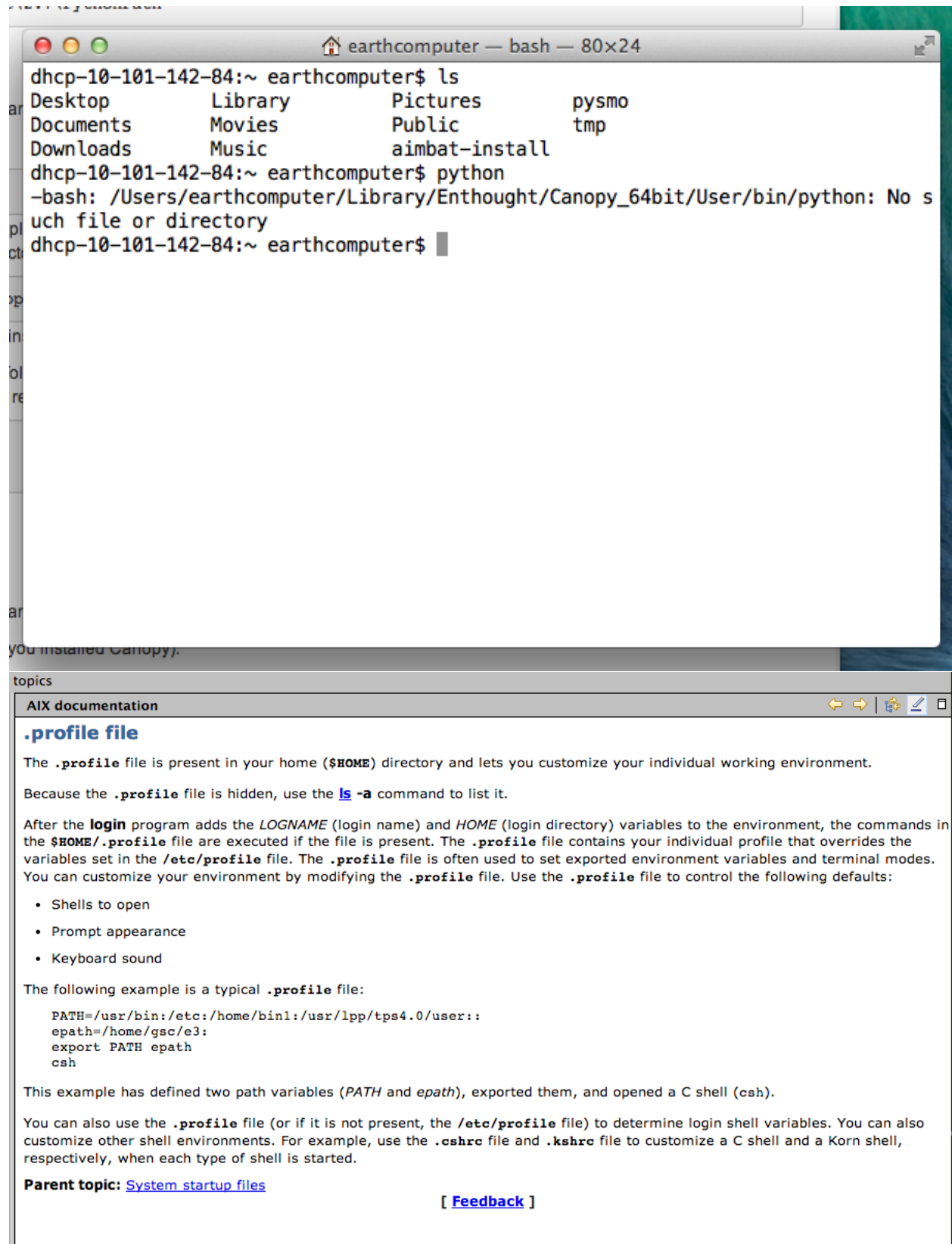
2.8.7 Path to python files not found

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*.

```
dhcp-10-101-142-84:~ earthcomputer$ ls
Desktop          Library          Pictures          pysmo
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 s
uch file or directory
dhcp-10-101-142-84:~ earthcomputer$ 
```

.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 exists.

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:

```
### computing environment setting for Linux and Mac

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

# add path from github directory
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"

###sourcing my scripts/programs
export PATH=$PATH:$SCRIPT
#export PATH=$PATH:$SCRIPT/gmt

###
export SVN_EDITOR=vim


".bashrc" 110L, 3266C
```
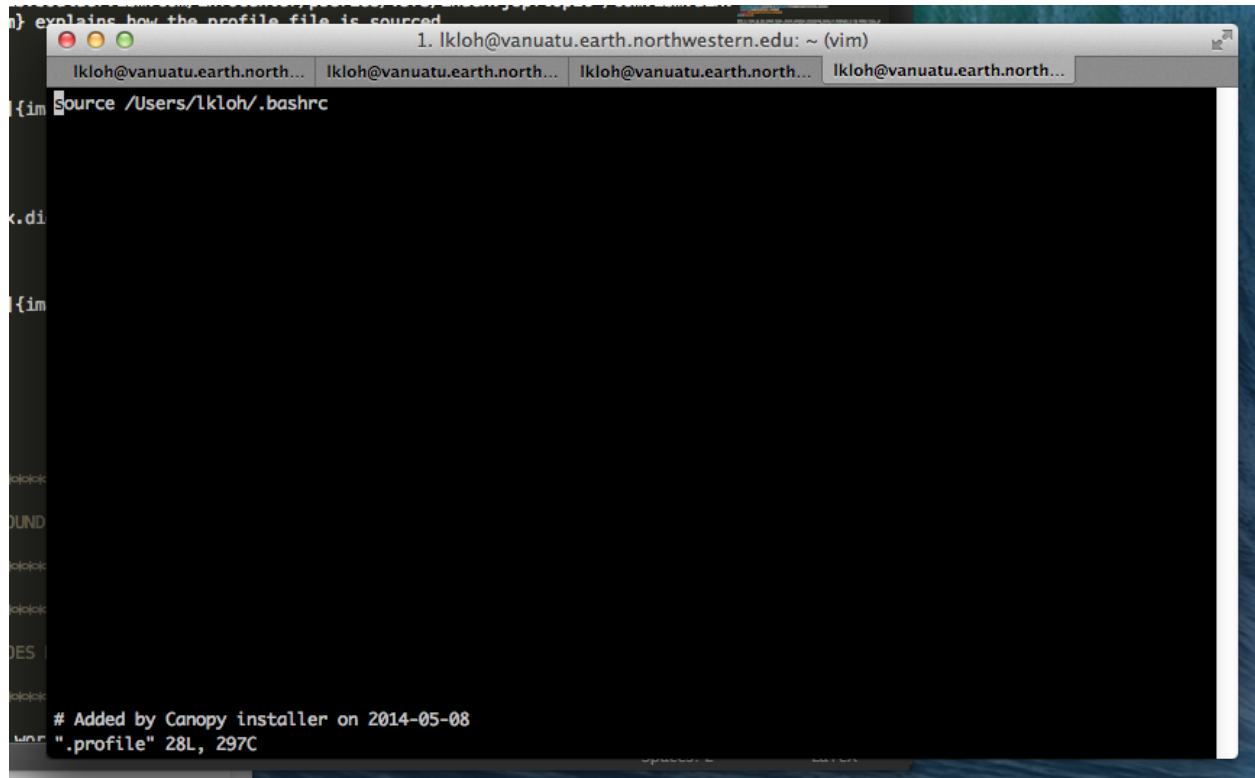
INSTALLING AIMBAT

3.1 Getting the Packages

AIMBAT is released as a sub-package of pysmo 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.

3.2 Where to install the packages

There are several options to install the packages. If you just want to use AIMBAT, it is best to store it somewhere where you would not touch the packages so easily, such as the Python `site-packages` directory. If you would like to make some changes to the Python code, it is best to store it somewhere pretty accessible, such as your `home` directory on your computer, or in `Documents`.

3.2.1 Installing into the Python Site-Packages Directory

To find out where the Python Site-Packages Directory is, in the python console, do:

```python
import site;
site.getsitepackages()
```

Whatever is output is obtained, lets call it `<pkg-install-dir>`. Make a directory called `pysmo`, and place the sac and aimbat directories inside `<pkg-install-dir>/pysmo`.

3.2.2 Installing into the home directory

Open your terminal. Type `open .` and that will open your home directory. Transfer the `aimbat` and `sac` repositories inside there.

3.3 Building the Pysmo Packages

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

3.3.1 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, this should happen after you type `from pysmo import sac`

3.3.2 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 directory the `aimbat` package was placed in, and type:

```
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.

Bash Shell Users: `export PATH=$PATH:<path-to-folder>/aimbat/scripts` in `.bashrc` files.

C Shell Users: `setenv PATH=$PATH:<path-to-folder>/aimbat/scripts` in `.bashrc` files.

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

GETTING DATA

There are several ways to obtain data to input to AIMBAT. If want to suggest other tools, please *contact the authors*.

4.1 Standing Order for Data
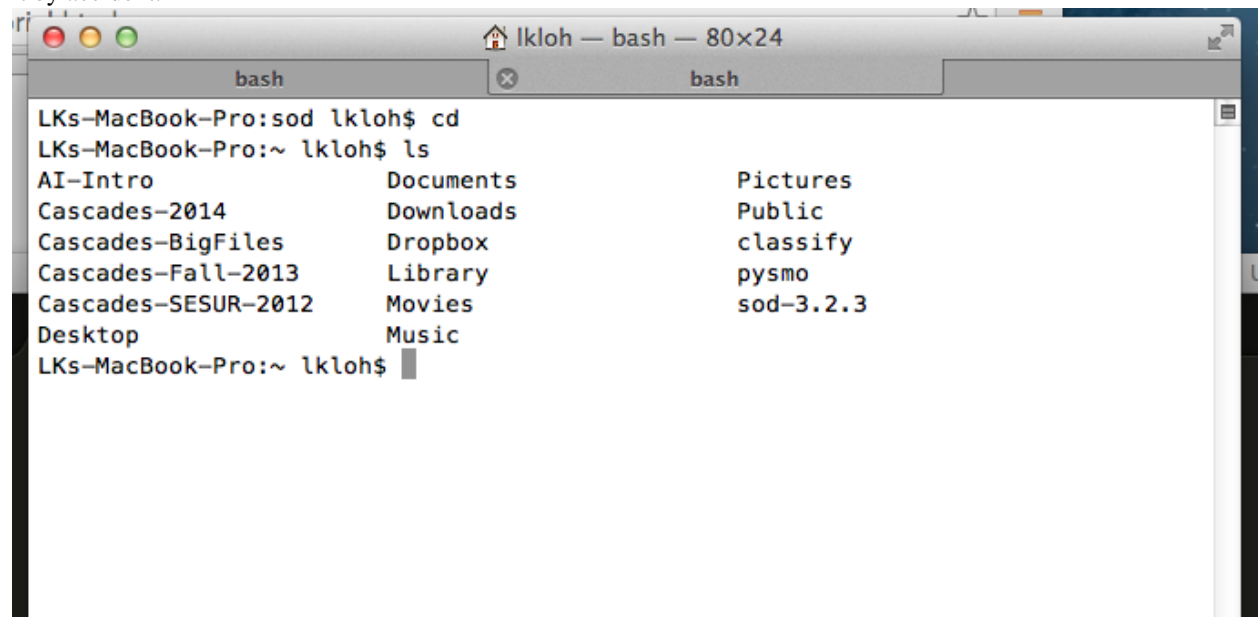
From the SOD website:

> 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.

4.1.1 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 its not too easy to delete it by accident.

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$
```
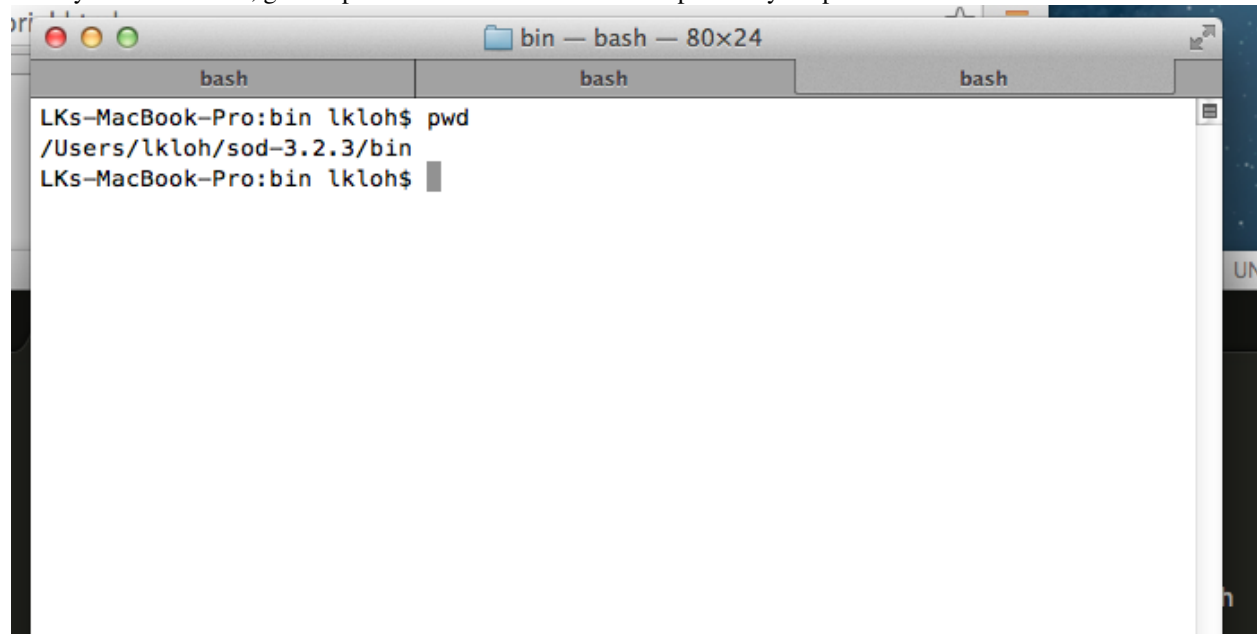
Inside my home directory's bash profile (you get the by typing *cd*), you put the path to *sod-3.2.3/bin* by adding in either the *bash* or *bash_profile* or *profile* files:

4.1.2 Downloading Data with SOD

> **Authors** Trevor Bollmann

1. **Create a sod recipe and place it in the folder that you would like the data to download to.**

    - `sod -f <recipename>.xml`

2. **Run `sodcut.sh` to cut the seismogram around phase wanted**

    - check model within `cutevseis.sh`

    - run using `sodcut.sh <name>`

    - watch `sdir = processed seismograms`

    - Run over the entire downloaded directory (the files sod downloaded)

3. **Run `sodpkl.sh` (converts *.sac* files to python pickles)**

    - run using `sodpkl.sh [options] <directory>`

    - output will automatically be zipped

    - run in DATA directory

4. **Run `ttpick.py` (does travel time picking with plotting)**

    - can use `iccs.py` but it does not have plotting capabilities

    - run using `ttpick.py [options] <pkl.gz file>`

    - do this one event at a time

    - use `sacp2` to look at the stacking of the seismograms

    - you can sort the seismograms using the `-s` flag

5. **run `getsta.py` (creates a `loc.sta` file)**

    - `getsta.py [options] <pkl.gz files>`

6. **Run EITHER of these:**

    - FIRST CHOICE

    run `mccc2delay.py` (converts mccc delays to actual delays) by doing `mccc2delay.py [option] <.mcp files>`

    run `getdelay.py` (creates a delay file) by doing *getdelay.py [options] <*.px>*. Can possibly use *do-plotsta.sh*, plots all of the events and their station delays

    Run `evmcdelay.sh`

    - SECOND CHOICE

    `ttcheck.py` to compare the delay times of the p and s waves. Should form a nice cloud with the mean value in line with the cloud.

7. **If you need to remove a station from an event you can use `pklsel.py`**

    - Run using `pklsel.py [pkl file] -d [stnm]` to remove one station
    - Only works for one event at a time

8. **If you need to filter the data to be able to pick use `evsacbp.sh`**

    - run using `evsacbp.sh [pkl file] bp1 bp2`
    - Automatically uses two corners
    - run in the whole downloaded directory (the one with the sac directory)

FIVE

FILTERING DATA

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

5.1 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.

5.2 Saving Options

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

Button Title	What it does
Save Headers Only	Saves the SAC headers only
Save Headers & Filter Parameters	Saves the SAC headers and filter parameters used. Those filter parameters will automatically the next time the SAC headers are loaded in AIMBAT
Save Headers & Override Data	Save SAV headers and write in the filtered data

ANALYZING DATA

6.1 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

7.1 Backend

Matplotlib works with six GUI (Graphical User Interface) toolkits: #. WX #. Tk #. Qt(4) #. FTK #. Fltk #. macosx

The GUI of AIMBAT uses the following to support interactive plotting: #. GUI neutral widgets #. GUI neutral event handling API (Application Programming Interface)

Examples given in this documentation are using the default toolkit `Tk` and backend `TkAgg`.

Visit these pages for an explanation of the backend and *how to customize it <http://matplotlib.org/users/customizing.html#customizing-matplotlib>*.

Put the following line in you ''matplotlibrc'' file::

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

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

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

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

3. a file specified by environment variable `TTCONFIG`

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

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' exibility. 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.

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.

8.1 Automated Phase Alignment

The ICCS algorithm calculates an array stack from predicted time picks, cross-correlates each seismogram with the array stack to Find the time lags at maximum cross-correlation, then use the new time picks to update the array stack in an iterative process. The MCCC algorithm cross-correlates each possible pair of seismograms and uses a least-squares method to calculate an optimized set of relative arrival times. Our method is to combine ICCS and 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 8.1: Time picks and their SAC headers used in the procedure for measuring teleseismic body wave arrival times.

Step	Algorithm Time Window	Input			Output	
			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.

## 8.2 Picking Travel Times

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

8.2.1 Getting into the right directory

In the terminal, cd into the directory with all 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 you 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.

8.2.2 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.
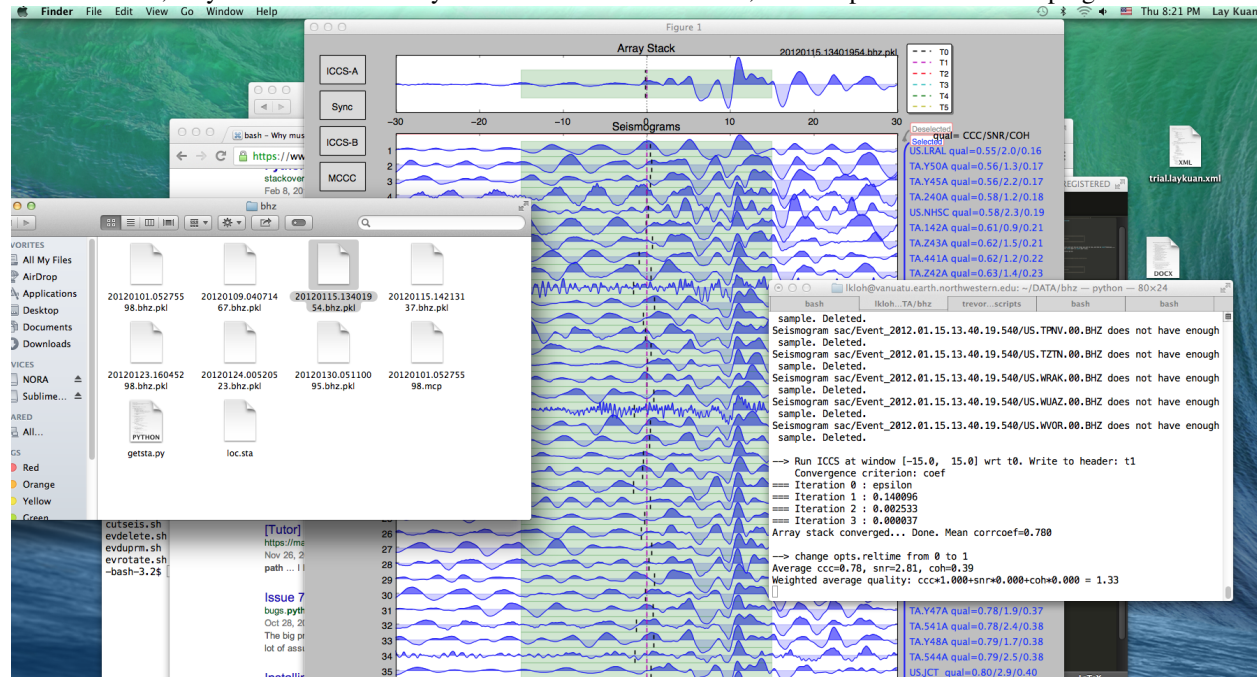
8.2.3 ICCC-A

`ICCC-A` 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.

8.2.4 Get rid of really 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, if there are more than enough seismograms, so it is safe to throw out any that deviate more than a bit from the array stack.

8.2.5 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.

8.2.6 ICCC-B

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

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

8.2.7 MCCC

Hit `MCCC` to run the Multi-Channel cross-correlation. Do not hit `ICCC-A` or `ICCC-B` 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.

8.2.8 Manually pick the arrival times using t2

For an earthquake, it is expected that the arrival times should be identical in an idealize 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 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. To zoom out, just hit `MCCC` again.

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.

### 8.2.9 SACP2 to check for outlier seismograms

Hit 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.

8.2.10 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 which to sort the seismograms in alphabetical order so that you can find the bad seismogrrams and correct them more easily. Hit the `sort` button and a window will popup for you to choose which sorting method to use. In this case, choose to sort the files by filename.
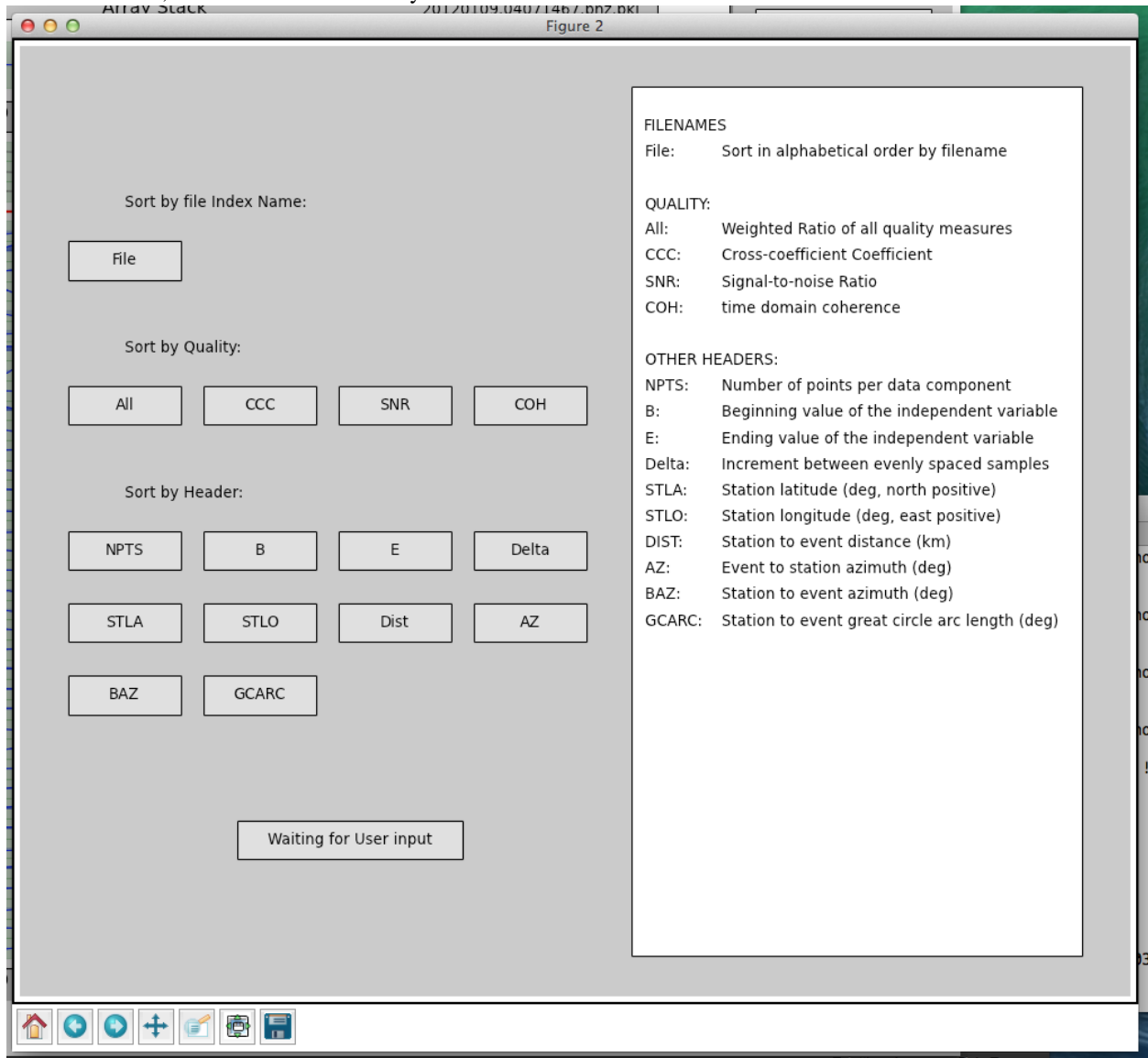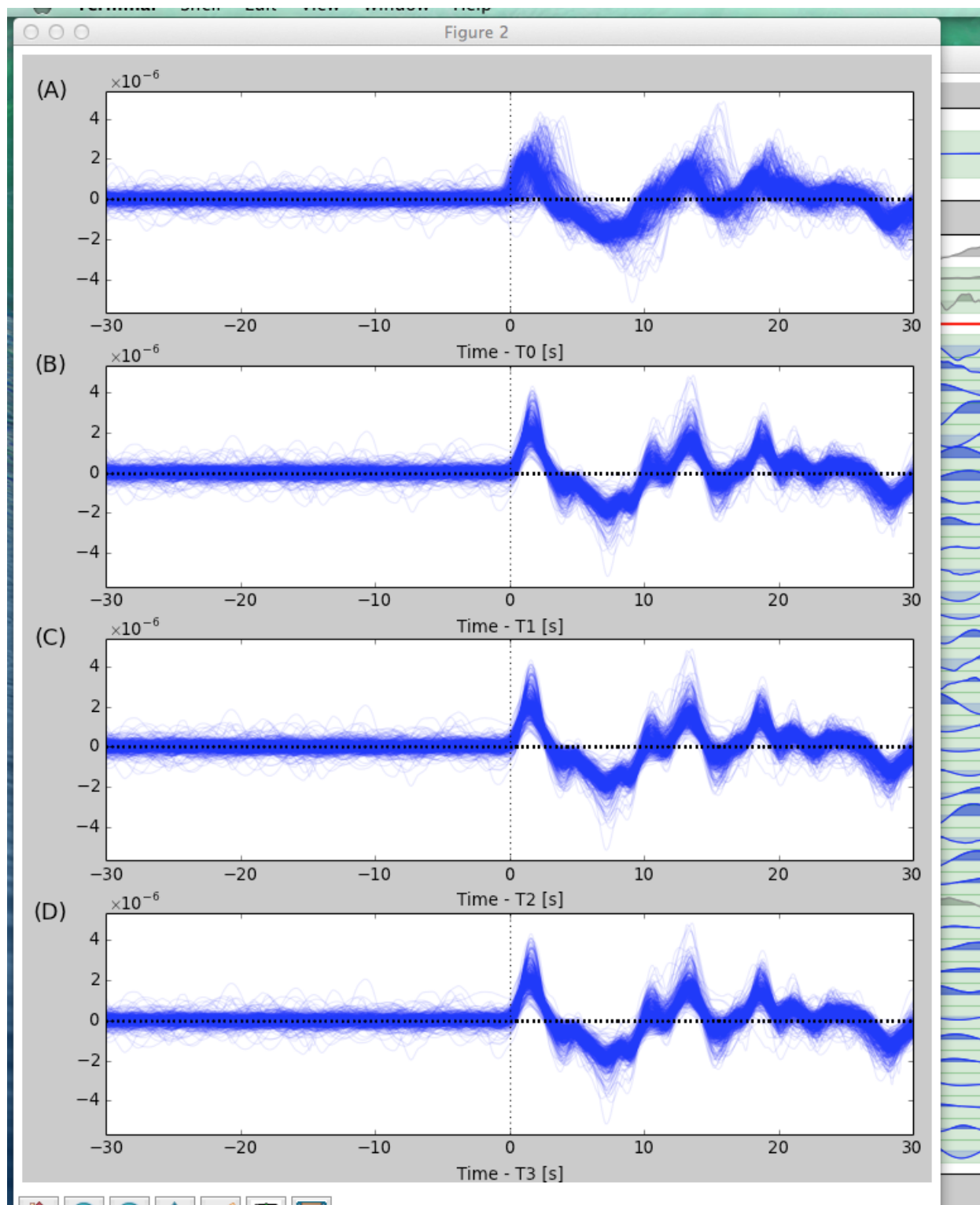
The seismograms are stretched to fit together, but they may be scaled differently.

8.3 What the Alignments Stand For

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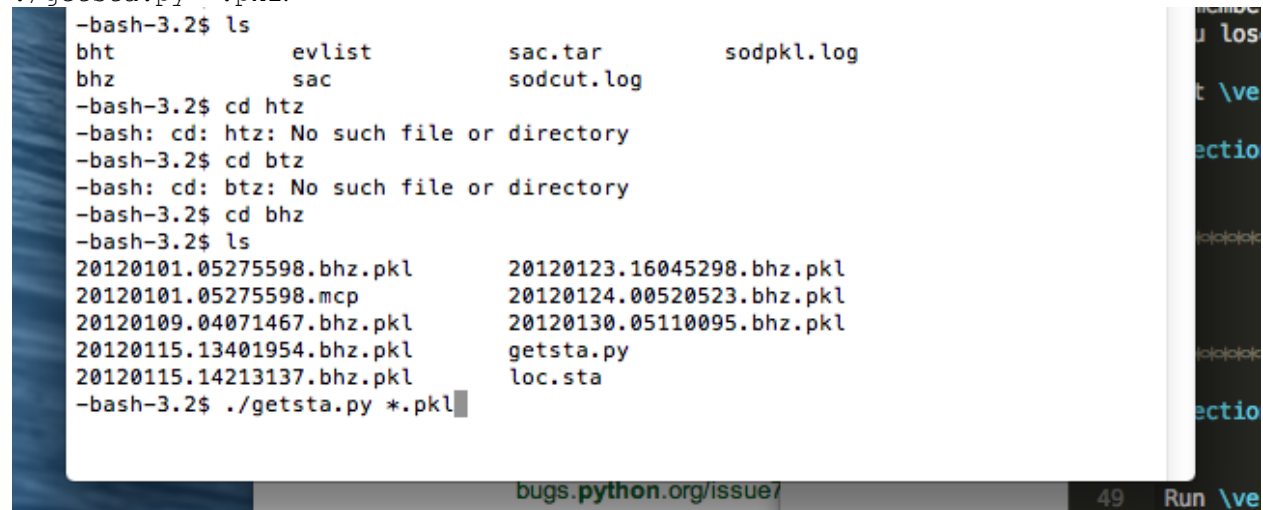- T3: MCCC pick
- T4: Zoom in

8.4 Post Processing

8.4.1 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.

8.4.2 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. You so this `./getsta.py *.pkl`.

8.4.3 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 so in your home directory 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!

8.4.4 Travel Times

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

UNIT TESTING

This section is mainly for those who which to make tweaks to AIMBAT themselves. We have added some unit tests to AIMBAT to ensure its robustness.

9.1 Running the Tests

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

```
python run_unit_tests.py
```

CITATIONS

ELEVEN

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

[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.

[Hunder2007] 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.