AIMBAT Documentation

Release 0.2.1

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

August 05, 2014

CONTENTS

Contents:


NORTHWESTERN UNIVERSITY
QUAECUMQUE SUNT VERA
1851
NORTHWESTERN
UNIVERSITY

CHAPTER

ONE

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 AT cmu DOT edu

- Xiaoting Lou

  Email: xlou AT u DOT northwestern DOT edu

- Suzan van der Lee

  Email: suzan AT earth DOT northwestern DOT edu

CHAPTER

TWO

INSTALLING 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 Github

Optional but recommended.

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.

Some users may already have Git installed, but if not, you can 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.

2.2 Macports

Macports will be needed to install some python packages. Download the package installer here.

2.3 Homebrew

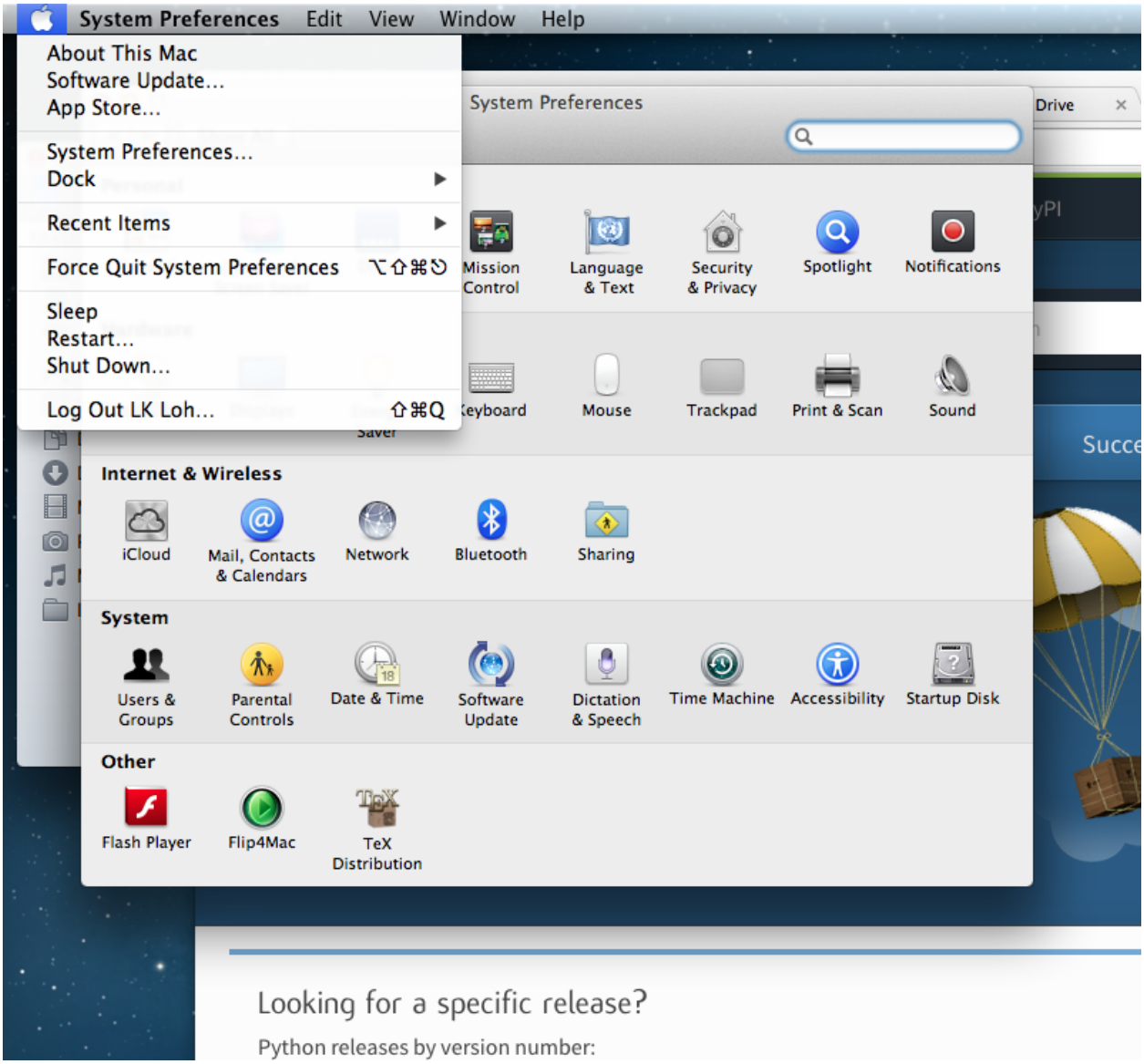
Homebrew will be needed to install some python packages. Get it by typing in your terminal:

```
ruby -e "$(curl -fsSL https://raw.github.com/Homebrew/homebrew/go/install)"
```

2.4 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.



2.5 Python Dependencies

Required for AIMBAT #. Numpy: Used for manipulating numbers and datasets #. Scipy: Used for data processing #. Matplotlib: Used for the majority of the plots in AIMBAT and the GUI

To check if you have python already, open the terminal and type `python`. If a console pops up, it should display the version of python you have. If not, the terminal will output:

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

If python is installed, you next need to check if the required packages are there. Open the python console by typing in the terminal:

```
python
```

Now, type:

```
import numpy
import scipy
import matplotlib
```

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

```
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.

2.5.1 Optional but recommended

iPython: Interactive console for python

2.6 Installing Basic Python Packages

2.6.1 Enthought Python

Strongly recommended.

Enthought Canopy is easy to install. Go to the website and download the free version of Express Enthought Canopy, which will give you the dependencies Numpy, Scipt, and Matplotlib.

2.6.2 If you cannot use Enthought Canopy ...

This section describes a possible way to install Python without using Enthough Canopy. It is *not* recommended and may cause problems on some systems, but the authors describe it just in case.

1. 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. 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 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

1. 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.

1. 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 (Python dependency)

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. 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
```

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 Macports

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

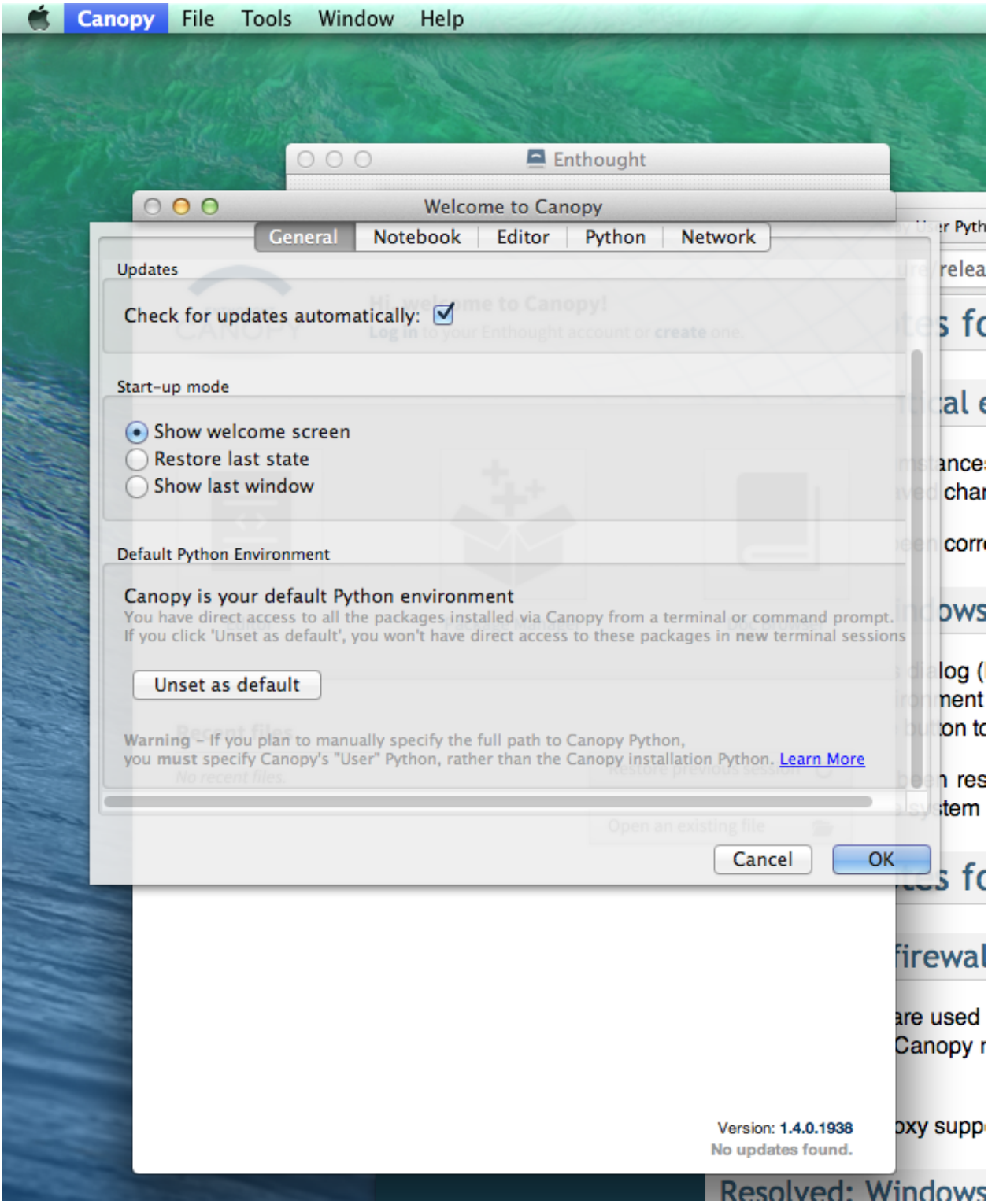
If ever the terminal commands such as ls stop working in the terminal, it could be that something went wrong with a path in the `.bashrc` or `.profile` files. If that happens you may not be able to open them in vim as that command would have stopped working as well. Instead, in the terminal, you do:

```
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.

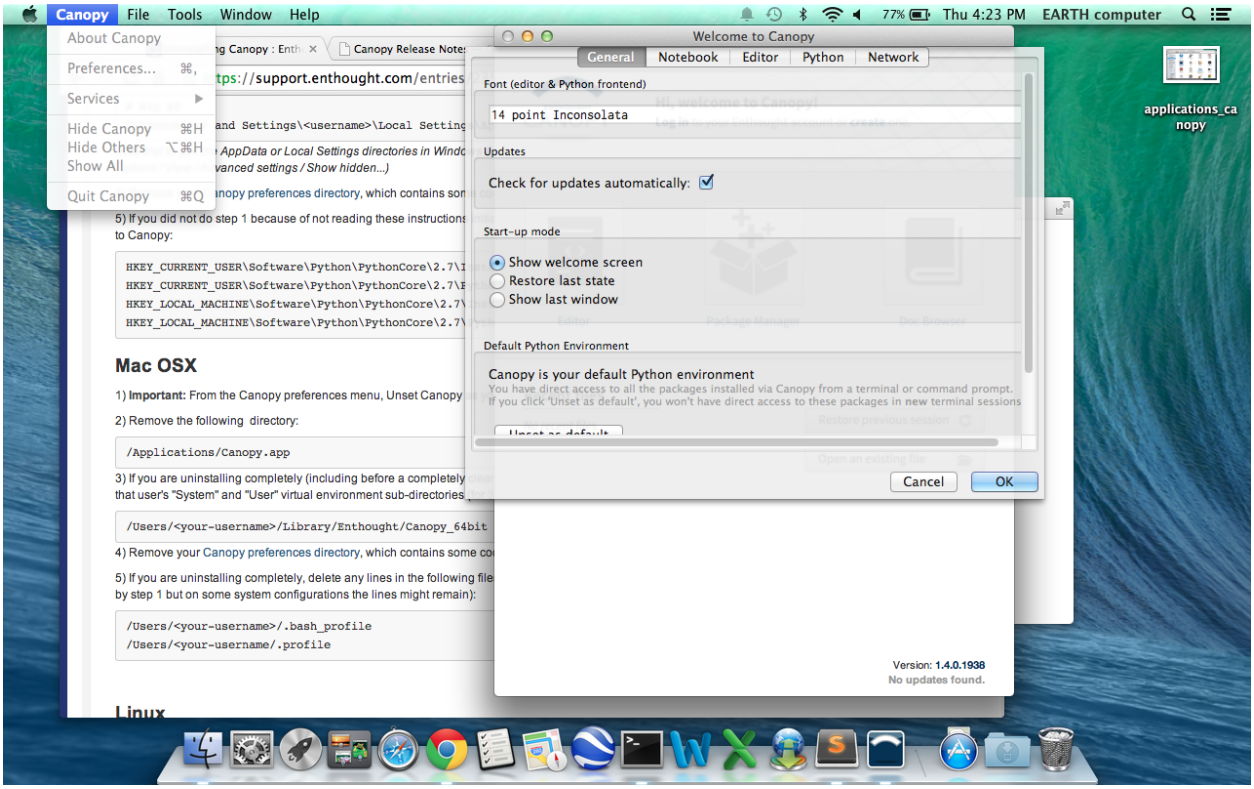
2.8.5 Installing Enthought Canopy

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.

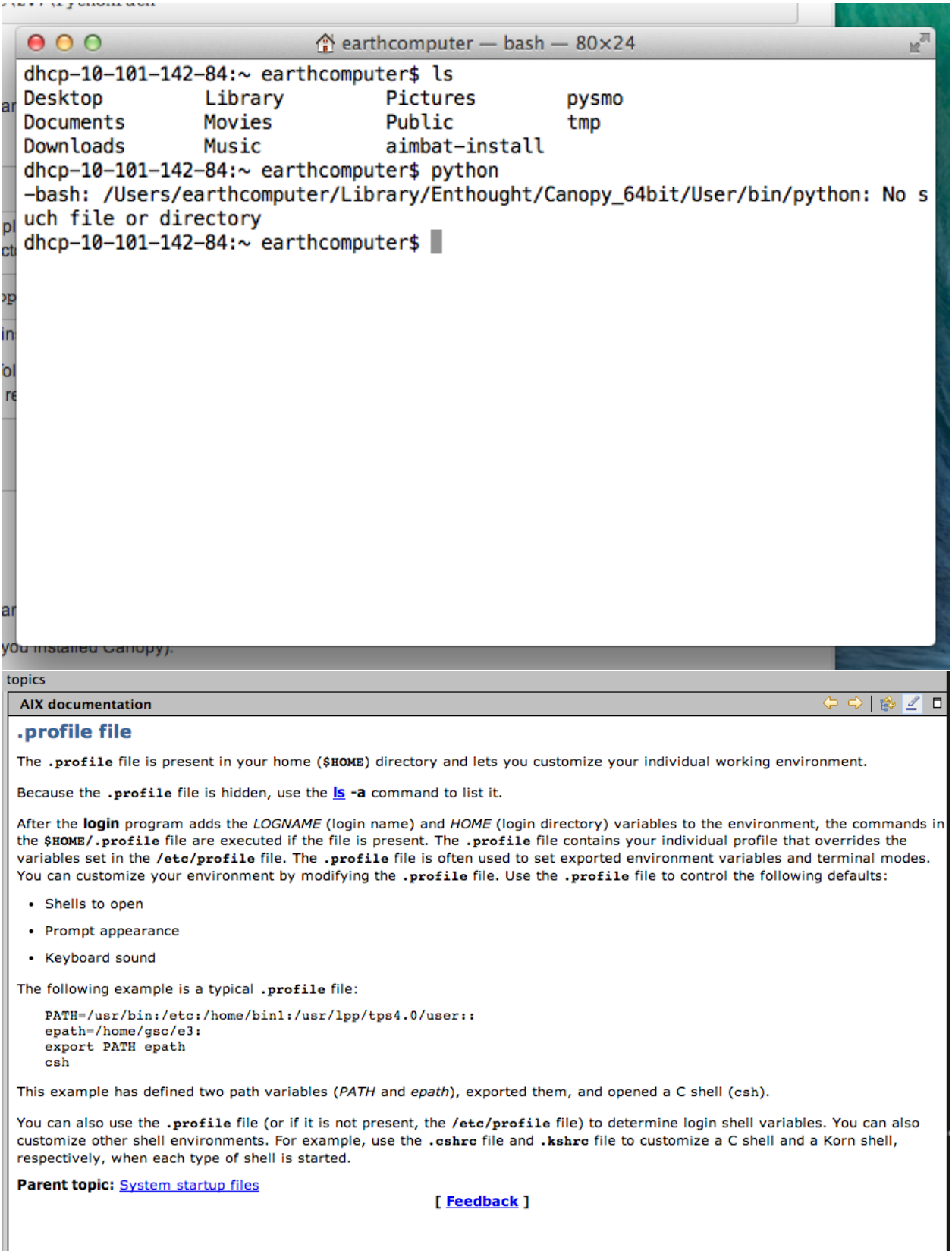


STEPS:

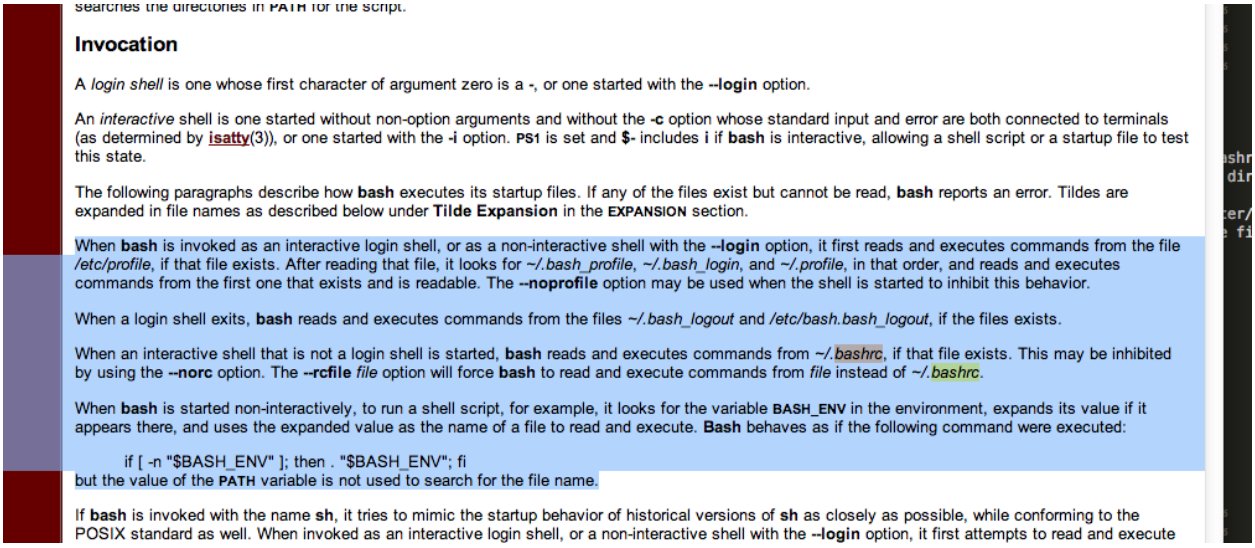
1. From the Canopy preferences menu, unset Canopy as your default Python.
2. For each Canopy user, delete the following directory which contains that user's "System" and "User" virtual environment subdirections.
3. Delete Canopy from the Applications folder.
4. Clean up the hidden files. Delete anything 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*.



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*.

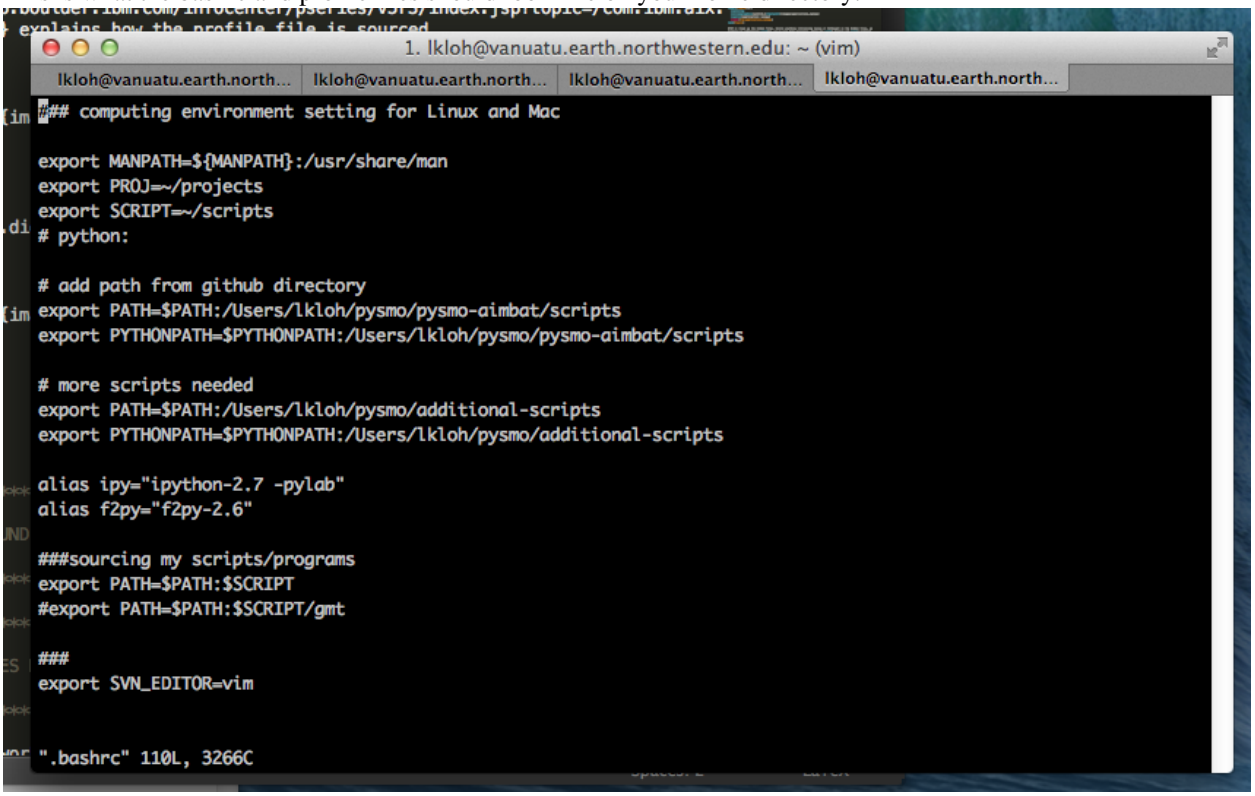
When **bash** is started non-interactively, to run a shell script, for example, it looks for the variable **BASH_ENV** in the environment, expands its value if it appears there, and uses the expanded value as the name of a file to read and execute. **Bash** behaves as if the following command were executed:

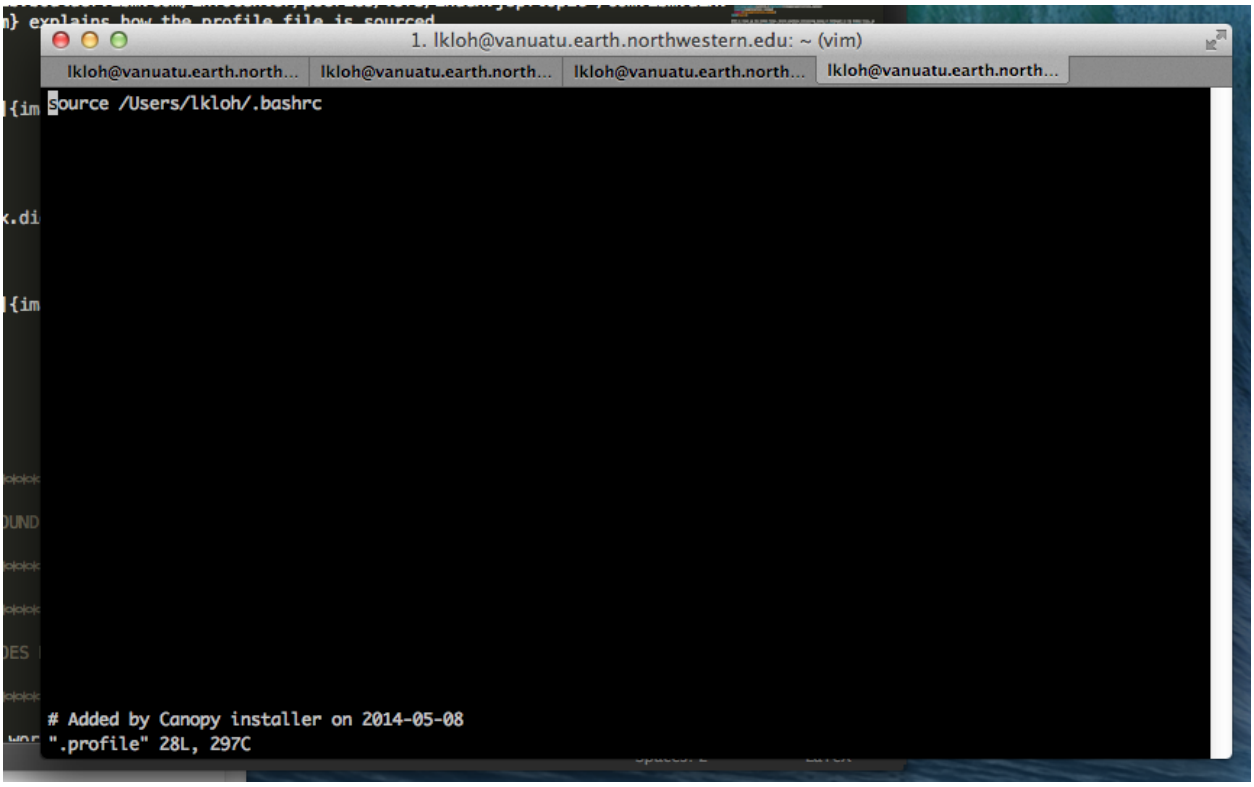
```
if [ -n "$BASH_ENV" ]; then . "$BASH_ENV"; fi
```

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:




1. lkloh@vanuatu.earth.northwestern.edu: ~ (vim)
lkloh@vanuatu.earth.north...
lkloh@vanuatu.earth.north...
lkloh@vanuatu.earth.north...
lkloh@vanuatu.earth.north...
source /Users/lkloh/.bashrc
Added by Canopy installer on 2014-05-08
".profile" 28L, 297C

CHAPTER

THREE

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.

You may want to download example code to run AIMBAT on, as well.

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:

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

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

3.4 Example Data

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

CHAPTER

FOUR

GETTING DATA

Note: Not necessary if you already have your own set ways of obtaining data. This section is added for completion.

There are several ways to obtain seismic data from IRIS to input to 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.

4.1 Obspy.fdsn for downloading data

4.1.1 Installing Obspy

We recommend using Macports to install Obspy as detailed in the *Installation* section here. If you have installed Enthough Canopy:

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

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

4.1.2 Did the installation work?

If 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, and type:

```
import obspy
```

If there are no errors, your installation has worked.

4.1.3 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.

4.2 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 was to download seismic data from IRIS, SOD allows for better fine-tuning of data obtained.

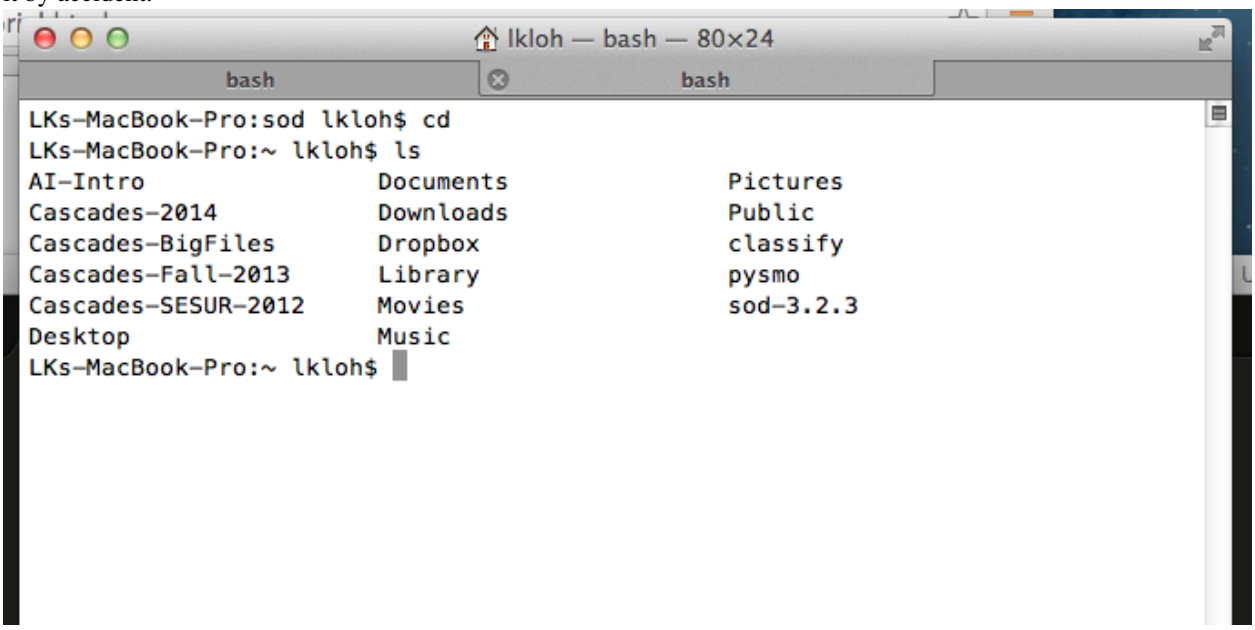
From the SOD website:

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.

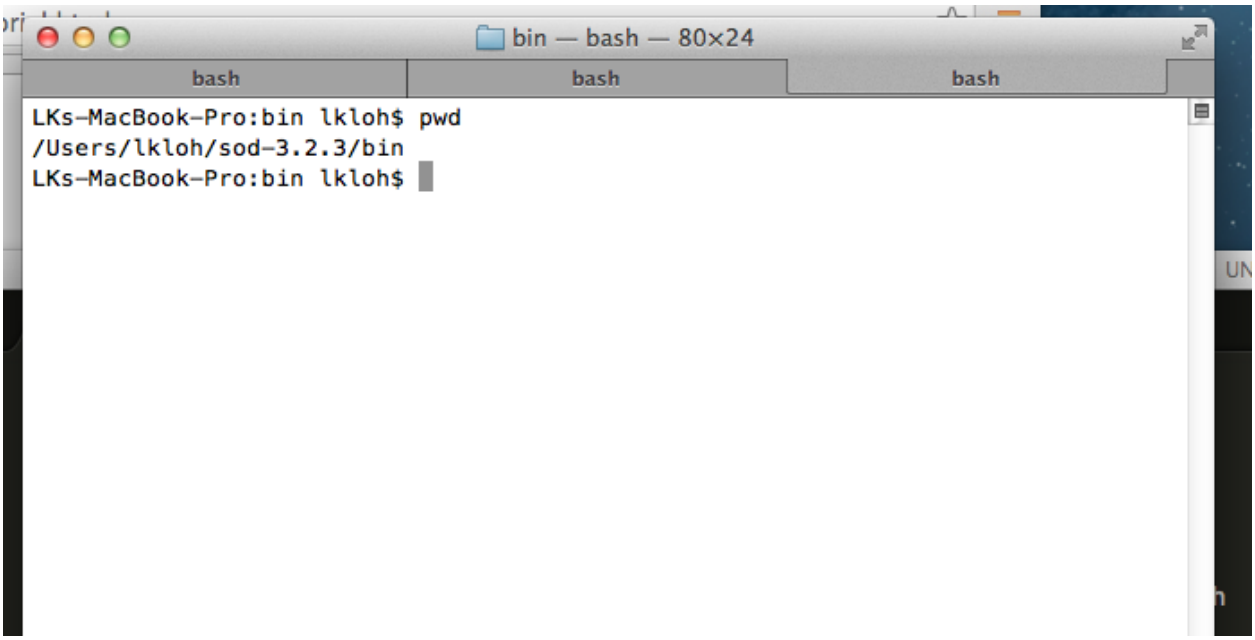
4.2.1 Installing SOD

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 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.



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.

4.2.2 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 container the filtered and unfiltered data.

CHAPTER

FIVE

SAC INPUT/OUTPUT PROCEDUES FOR AIMBAT

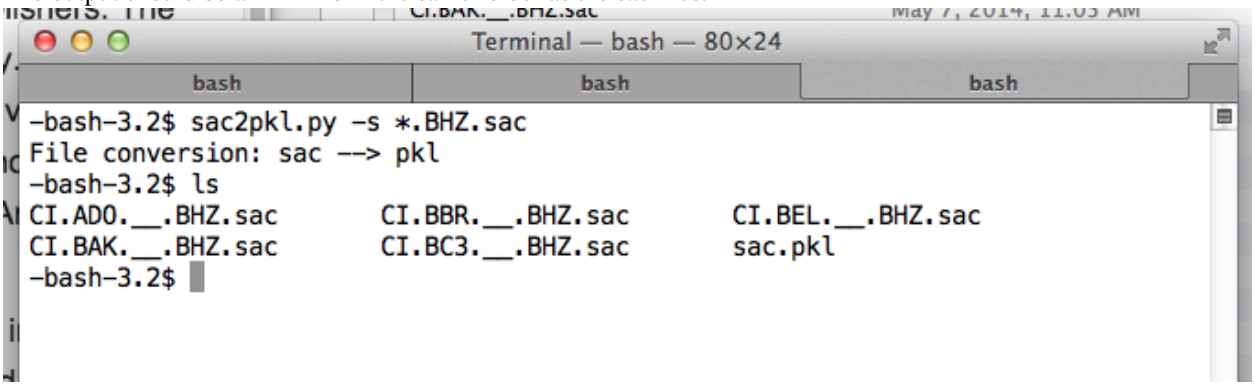
Once you have downloaded

5.1 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.



CHAPTER

SIX

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.

6.1 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.

6.2 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

CHAPTER

SEVEN

ANALYZING DATA

7.1 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.

CHAPTER

EIGHT

PARAMETER CONFIGURATION

8.1 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)

AIMBAT uses 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>*.

8.2 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' 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.

8.2.1 Example of AIMBAT configuration file *ttdefaults.conf*

ttdefaults.conf	Description
[sacplot]	
colorwave = blue	Color of waveform
colorwavedel = gray	Color of waveform which is deselected
colortwfill = green	Color of time window fill
colortwsele = red	Color of time window selection
alphatwfill = 0.2	Transparency of time window fill
alphatwsele = 0.6	Transparency of time window selection
npick = 6	Number of time picks (plot picks: t0-t5)
pickcolors = kmrcgyb	Colors of time picks
pickstyles	Line styles of time picks (use second one if ran out of color)
figsize = 8 10	Figure size for *plotphase.py*
rectseis = 0.1 0.06 0.76 0.9	Axes rectangle size within the figure
minspan = 5	Minimum sample points for SpanSelector to select time window
srate = -1	Sample rate for loading SAC data. Read from first file if srate < 0

[sachdrs]	
twhdrs = user8 user9	SAC headers for time window beginning and ending
ichdrs = t0 t1 t2	SAC headers for ICCS time picks
mchdrs = t2 t3	SAC headers for MCCC input and output time picks
hdrsel = kuser0	SAC header for seismogram selection status
qfactors = ccc snr coh	Quality factors: cross-correlation coefficient, signal-to-noise ratio, time domain coherence
qheaders = user0 user1 user2	SAC Headers for quality factors
qweights = 0.3333 0.3333 0.3333	Weights for quality factors

[iccs] or Align/Refine	
srate = -1	Sample rate for loading SAC data. Read from first file if srate < 0
xcorr_modu = xcorrf90	Module for calculating cross-correlation: xcorr for Numpy or xcorrf90 for Fortran
xcorr_func = xcorr_fast	Function for calculating cross-correlation
shift = 10	Sample shift for running coarse cross-correlation
maxiter = 10	Maximum number of iteration
convepsi = 0.001	Convergence criterion: epsilon
convtype = coef	**Type of convergence criterion: coef for correlation coefficient,** or resi for residual
stackwgt = coef	Weight each trace when calculating array stack
fstack = fstack.sac	SAC file name for the array stack

[mccc]	
srate = -1	Sample rate for loading SAC data. Read from first file if srate < 0
ofilename = mc	Output file name of MCCC.
xcorr_modu = xcorrf90	Module for calculating cross-correlation: xcorr for Numpy or xcorrf90 for Fortran
xcorr_func = xcorr_faster	Function for calculating cross-correlation
shift = 10	Sample shift for running coarse cross-correlation
extraweight = 1000	Weight for the zero-mean equation in MCCC weighted lsqr solution
lsqr = nowe	Type of lsqr solution: no weight
#lsqr = lnco	Type of lsqr solution: weighted by correlation coefficient, solved by lapack
#lsqr = lnre	Type of lsqr solution: weighted by residual, solved by lapack
rcfile = .mcccrc	Configuration file for MCCC parameters (deprecated)
evlist = event.list	File for event hypocenter and origin time (deprecated)

signal	
tapertype = hanning	Taper type
taperwidth = 0.1	Taper width

CHAPTER

NINE

SAC DATA ACCESS

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

9.1.1 *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 refered to as <*example-event-dir*>.

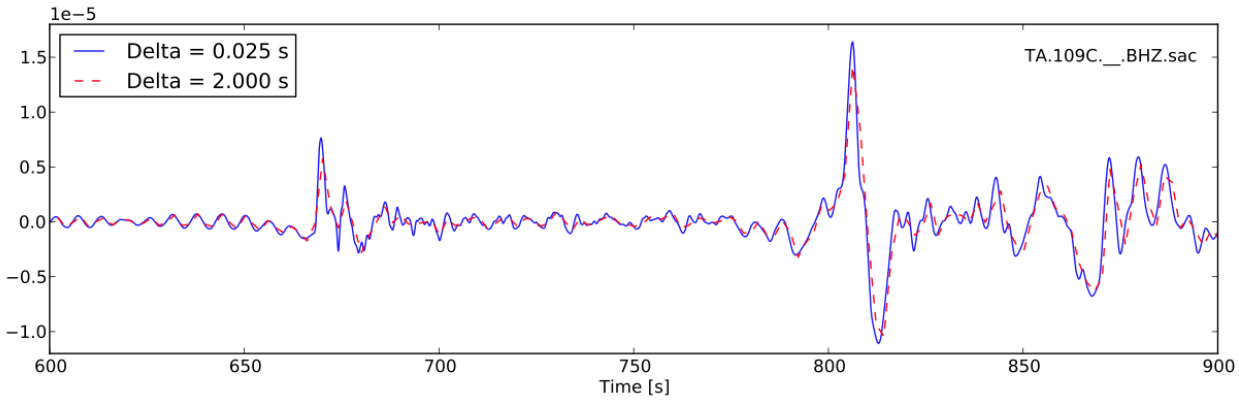
```
from pysmo.sac.sacio import sacfile
from numpy import linspace, array
from scipy import signal
import matplotlib.pyplot as plt
import matplotlib.transforms as transforms

# 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()
```

9.1.2 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
```



9.2 Python Pickle for SAC Files

The `pysmo.sacio` module converts SAC files to `sacfile` objects. Any modification of the objects are instantly written to files. In data processing, the user may want to abandon changes made earlier, which brings the need of a buffer for the `sacfile` objects.

The `SacDataHdrs` class in the `pysmo.aimbat.sacpickle` module is written on top of `pysmo.sacio` to serves this purpose by reading a SAC file and returning a `sacdh` object that is very similar of the `sacfile` object. Essentially, the `sacdh` object is a copy of the the `sacfile` object in the memory, except that SAC headers 't0-t9', 'user0-user9', 'kuser0-kuser2' are saved in three Python lists.

A `gsac` object of the `SacGroup` class consists of a group of `sacdh` objects from event-based SAC data files, earthquake hypocenter information and station locations. An additional step is required to save changes in the `gsac` object to files.

In order to avoid frequent SAC file I/O, the `pickle/cPickle` module is used for serializing and de-serializing the `gsac` object structure. Thus the data processing efficiency is improved because reading and writing of SAC files are done only once each before and after data processing. Script `sac2pkl.py` does the conversions between SAC files and Python pickles.

Its usage message can be printed out by running 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.

..image:: SACdataAccess/help-sac2pkl

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.

9.3 SAC Plotting and Phase Picking

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

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

SAC plotting and phase picking functionalities are replicated and enhanced based on the GUI neutral widgets (such as Button and SpanSelector) and the event (keyboard and mouse events such as `key\_press\_event` and `mouse\_motion\_event` handling API of Matplotlib.

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

```
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()
```

9.3.1 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 which includes multiple SAC files. For example, a `bhz.pkl` file is generated from 22 vertical component seismograms `TA.[1-K]*Z` by running:

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

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

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

or:

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

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

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

..image:: SACdataAccess/egplot.png

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

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

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

9.4 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 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, show in the figure below.

..image:: SACdataAccess/sacppk.png

Another major improvement involves quality control with convenient operations to (de)select seismograms. In the GUI in 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 MINNSEL, --minnsel=MINNSEL
                        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, --ofilename=OFILENAME
                        Output file name. Default is $evdate.mc$phase
  -a, --allseis         Use all seismograms. Default to use selected ones.
```

CHAPTER

TEN

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.

10.1 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 10.1: 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.

10.2 Picking Travel Times

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

10.2.1 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.

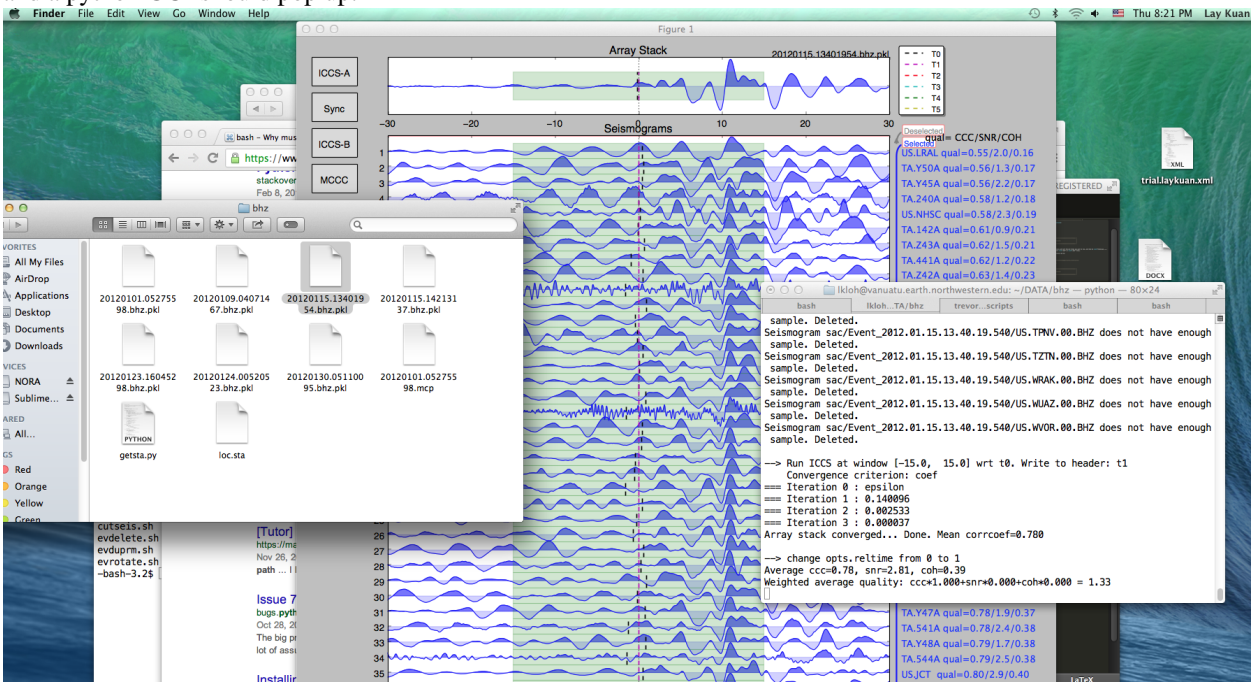
10.2.2 Running ttpick.py

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.



10.2.3 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.

10.2.4 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.

10.2.5 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.

10.2.6 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.

10.2.7 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.

10.2.8 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.

10.2.9 SACP2 to check for outlier seismograms

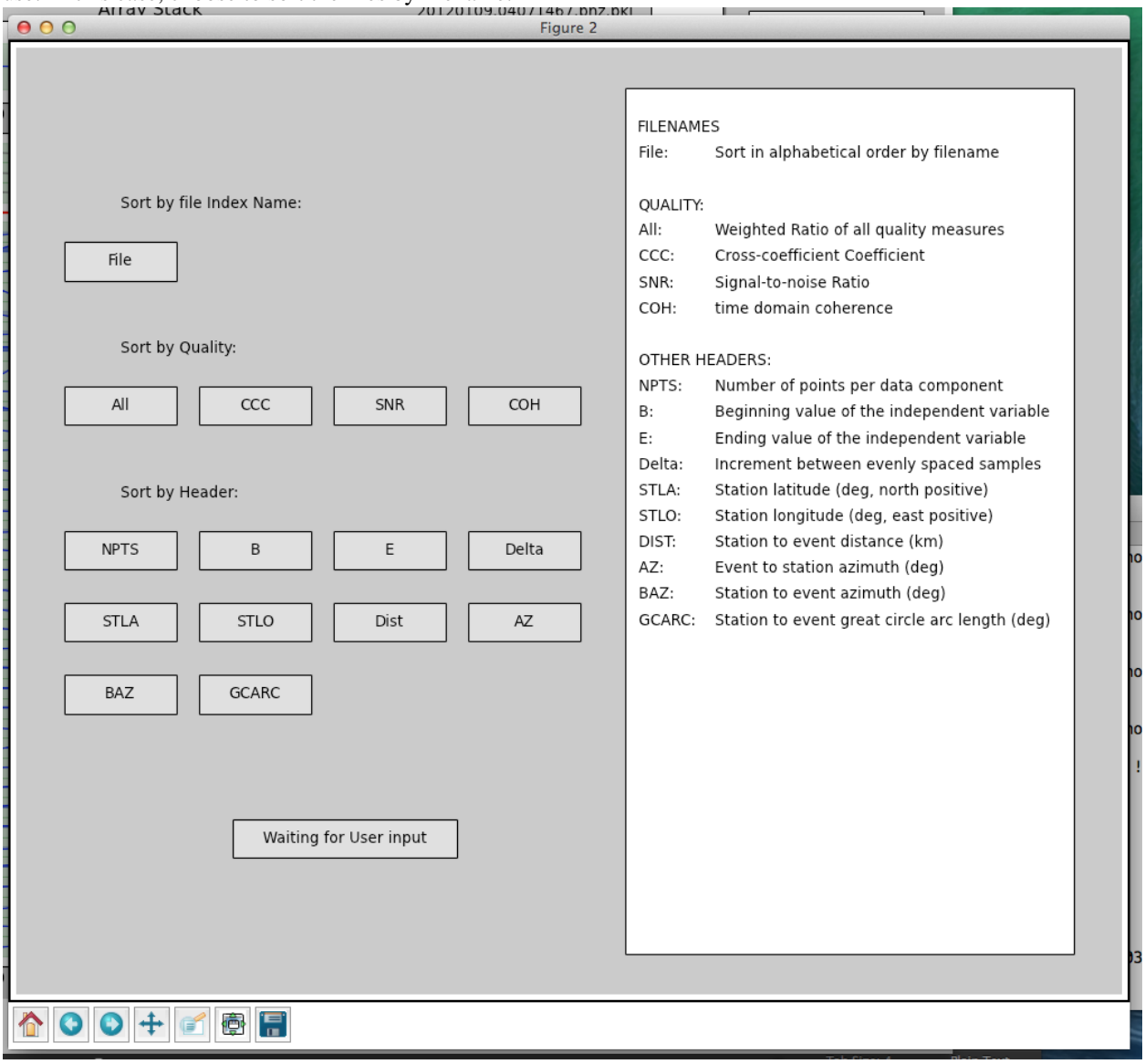
Hit and go to the last figure, (d). Zoom in to have a better look. Zooming in doesn't always work well; close and reopen the `SACP2` window if there are problems.

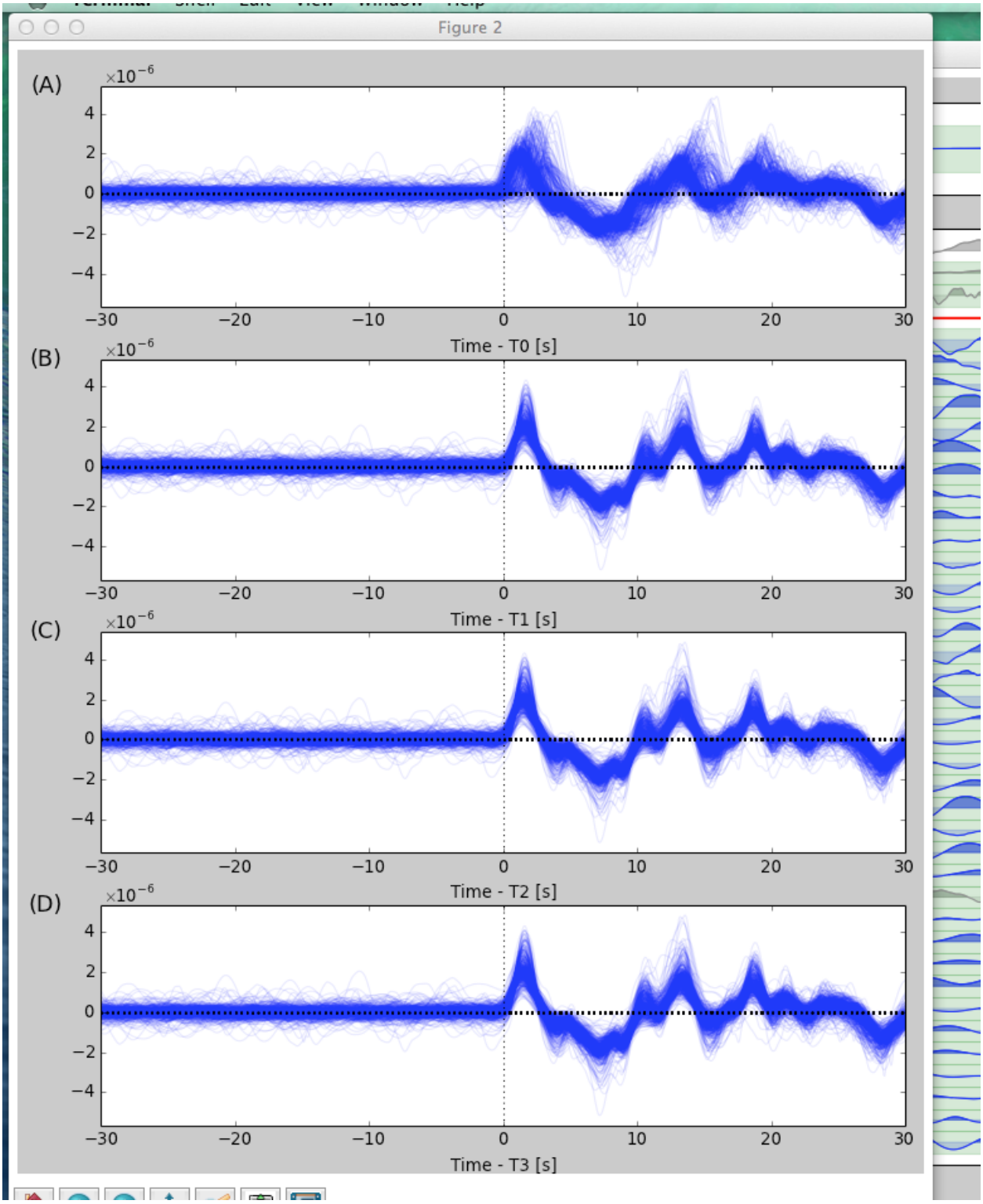
Click on the outliers that stray from the main group of stacked seismograms. The terminal will output the names of the seismograms that you clicked on, so you can return to the main GUI window and readjust the travel times.

10.2.10 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.

10.3 What the Alignments Stand For

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

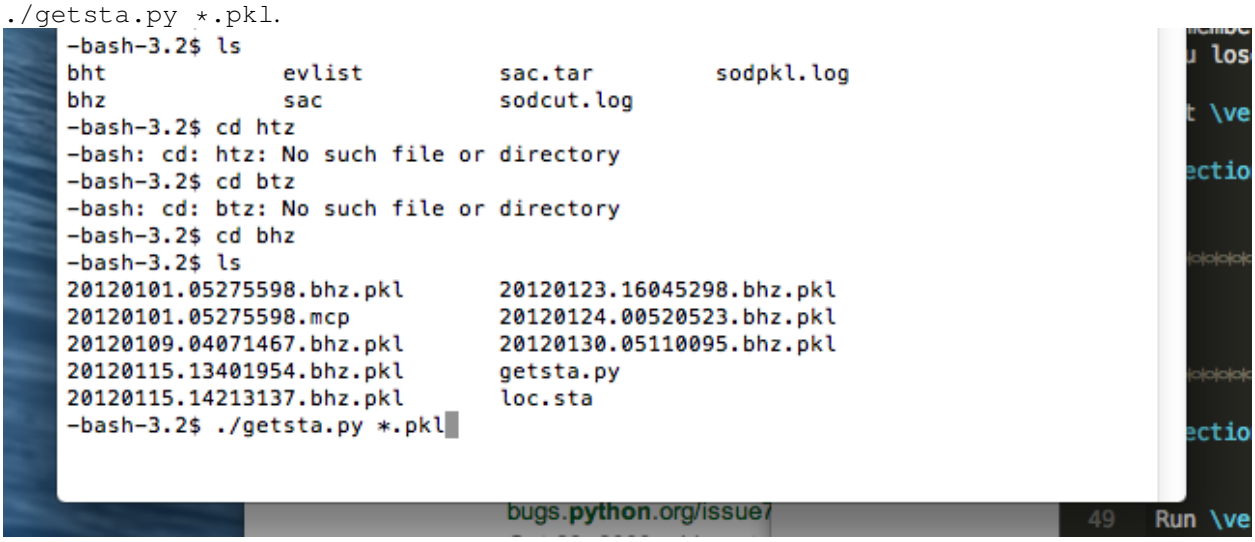
10.4 Post Processing

10.4.1 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.

10.4.2 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`.



10.4.3 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!

10.4.4 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.

CHAPTER

ELEVEN

VISUALIZING STATIONS ON A MAP

11.1 Coloring stations by delay times

- On the terminal, cd into the same folder where you sourced the SAC files to run `ttpick.py`, after running `MCCC` on a pickle file name, for instance, `earthquake-event.pkl`, you will get a mcp file named `earthquake-event.mcp`.
- Run `mccc2delay.py earthquake-event.mcp` and you will get a file with travel times called `earthquake-event.dtp`.
- Run `doplotsta.sh` and you will get a two `gif` files of the stations colored by delay times.

11.2 Computing delay times

The equation used to compute delays is explained here [BulandChapMan1983].

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 mcp files containing the results of picking travel wave arrival times on a set of seismograms.

Run *getsta.py* on the PKL file to get the file *loc.sta* containing the station locations. For example, in *data-example/example_pkl_files* type:

```
getsta.py 20120101.05275598.bhz.pkl
```

This will output a file, *loc.py*.

Next, run *mccc2delay.py* to convert the MCCC delays into actual delays. For example, in *data-example/example_pkl_files* type:

```
mccc2delay.py 20120101.05275598.bhz.pkl
```

You will get a *.px* file.

To create a delay file, run:

```
getdelay.py 20120101.05275598.bhz.pkl
```

To plot the stations colored by delay times, run *doplotsta.sh <file-name>.px*. For example, in *data-example/example_pkl_files* type:

```
doplotsta.sh 20120101.05275598.px
```

CHAPTER

TWELVE

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. See the Python Unit Testing Framework for more details.

12.1 Running the Tests

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

```
python run_unit_tests.py
```

CHAPTER

THIRTEEN

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.

13.1 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.

13.2 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 two commands builds the html for the webpage, while the last command makes a pdf version of the online documentation.

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

CHAPTER

FOURTEEN

CITATIONS

CHAPTER

FIFTEEN

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.

[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.