AIMBAT Documentation

Release 0.1.2

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

May 22, 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`

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

CHAPTER

TWO

INSTALLING DEPENDENCIES

2.1 Getting your operating system

You may need to know .. image:: installing-images/system_preferences.png

2.2 Installing Python

Shaowei Lin suggested Enthought Canopy to install all the Python packages easily. If you download the free version of 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 to use Macports or Pip.

2.3 Python Dependencies

- Numpy
- Scipy
- Matplotlib
- iPython (optional)

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 releases of `pysmo.sac` and `pysmo.aimbat` are available for download at the official project webpage and Github.

The packages should be installed into the Python site-packages directory. To find out where that is, in the python console, do:

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

Whatever is output there, lets call it `<pkg-install-dir>`. You can choose to install AIMBAT either locally or globally, depending on whether you want all users of the computer to have access to it.

Make a directory called `pysmo`, and place the sac and aimbat directories there.

Now that we know the location of the site-packages direction, cd into it. Call the path to it `<pkg-install-dir>`. Notice that in this case, the site-packages has been installed for all users on the computer, not just the current user's home directory.

Put the two Python packages inside the directory.

3.2 Installing pysmo.sac

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

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

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

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

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

This will install it to your home directory only, not for all users on the computer. Try not to use this option though, as installing without the `sudo` command has caused problems in the past.

If you successfully installed the sac module, in the python console, this should happen after you type `from pysmo import sac`

3.3 Installing pysmo.aimbat

Three sub-directories are included in the <pkg-install-dir>/pysmo/pysmo-aimbat-0.1.2> directory: example, scripts, and src, which contain example SAC files, Python scripts to run at the command line, and Python modules to install, respectively.

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

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

In the directory <pkg-install-dir>/pysmo/pysmo-aimbat-0.1.2, type:

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

to install the src directory.

Add <pkg-install-dir>/pysmo/pysmo-aimbat.0.1.2/scripts to environment variable PATH in a shells start-up file for command line execution of the scripts.

Bash Shell Users: export PATH=$PATH:<pkg-install-dir>/pysmo/pysmo-aimbat-0.1.2/scripts in .bashrc files.

C Shell Users: setenv PATH=$PATH:<pkg-install-dir>/pysmo/pysmo-aimbat-0.1.2/scripts in .bashrc files.

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

CHAPTER

FOUR

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

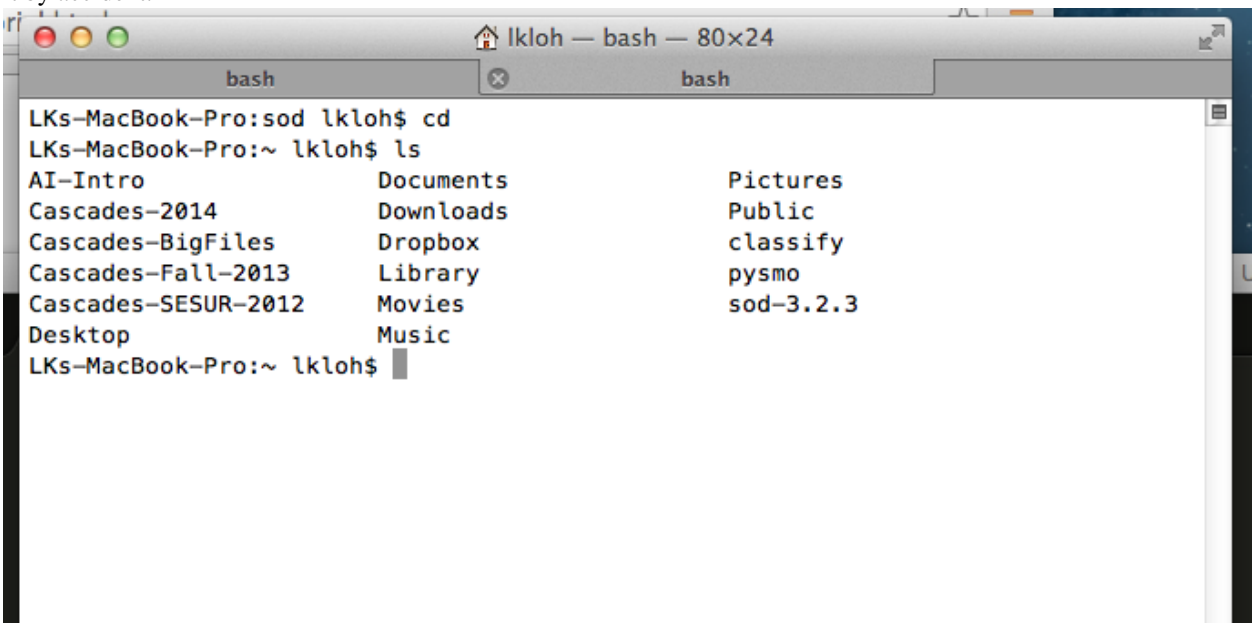
From the SOD website:

> 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.

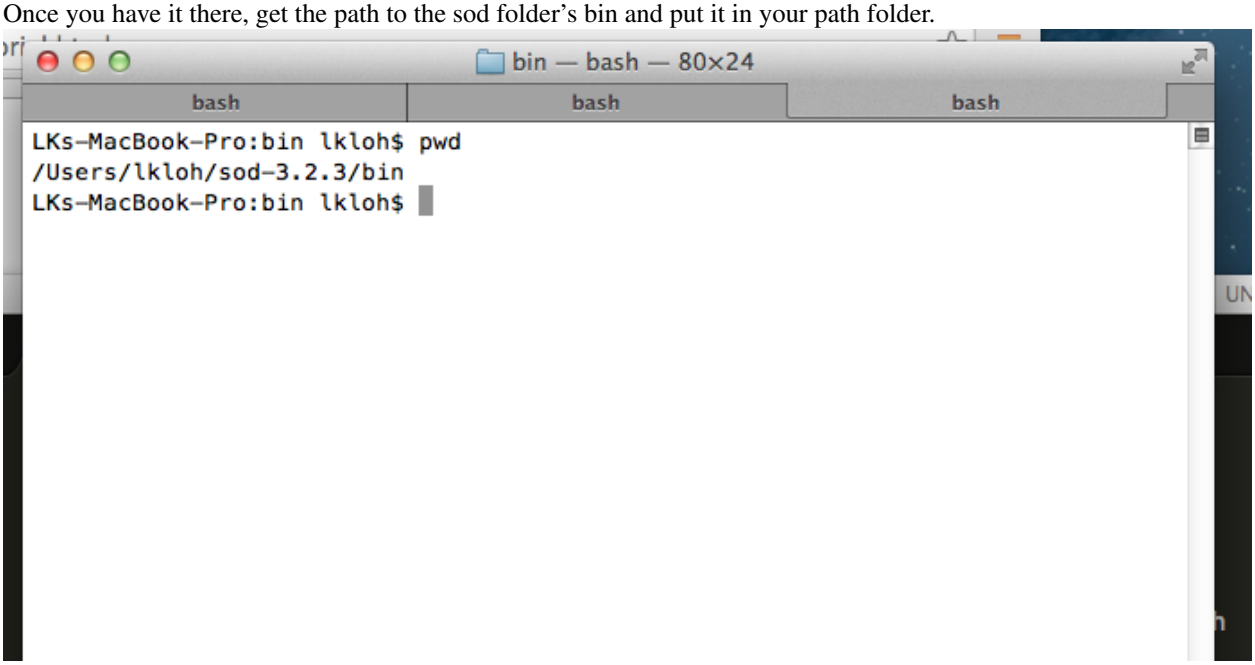
4.1.1 Installing SOD

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.



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 *doplotsta.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)

CHAPTER

FIVE

ANALYZING DATA

5.1 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

SIX

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.

6.1 Automated Phase Alignment

The ICCS algorithm calculates an array stack from predicted time picks, cross-correlates each seismogram with the array stack to Find the time lags at maximum cross-correlation, then 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 6.1: 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.

6.2 Picking Travel Times

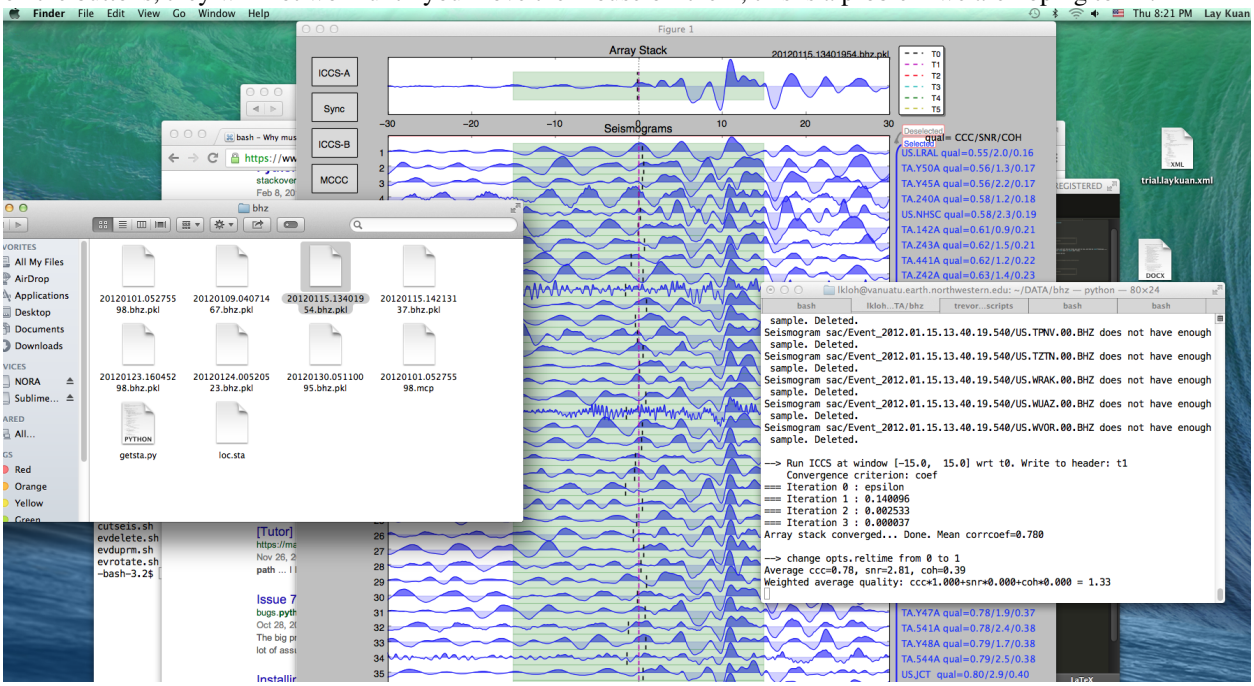
This section explains how to run the program `ttpick.py` to get the travel times you want.

6.2.1 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.

6.2.2 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.



6.2.3 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.

6.2.4 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. If you don't filter your data, you'll have to throw out more seismograms.

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

6.2.5 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.

6.2.6 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.

6.2.7 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.

6.2.8 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.

6.2.9 Go through the badly aligned seismograms and realign the travel times manually

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

However, you may which to sort the seismograms in alphabetical order so that you can find the bad seismogrrams and correct them more easily. Run:

```
ttpick.py -s -i ___.bhz.pkl
```

and scroll through the pages. Notice that clicking through the pages may be slow, move the mouse around and off/on the GUI window to stop it stalling. You can also hit `MCCC` to jump back to the front page.

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

6.3 What the Alignments Stand For

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

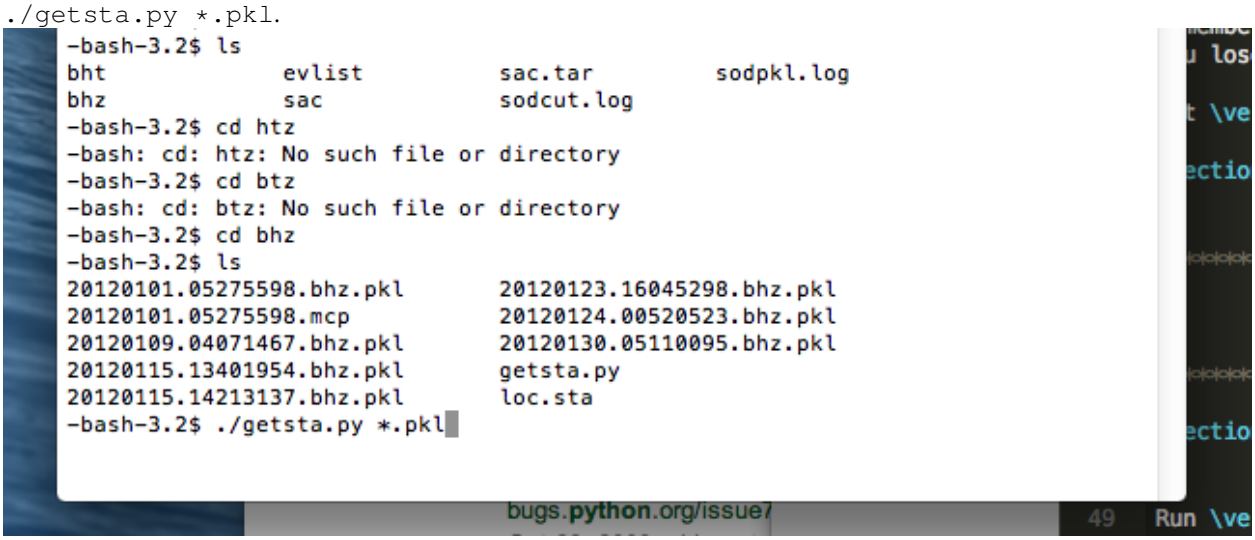
6.4 Post Processing

6.4.1 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.

6.4.2 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`.



CHAPTER

SEVEN

CITATIONS

CHAPTER

EIGHT

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 & Engi- neering, 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.