PhasePApy

A Robust Pure Python Package for

Automatic Identification of Seismic Phases

User Manual

Version 1.1

February 2016

by

Chen Chen

c.chen@ou.edu; c.chen8684@gmail.com

University of Oklahoma

Oklahoma Geological Survey

Austin Holland

austin.holland@ou.edu; aaholland@usgs.gov

Oklahoma Geological Survey

U.S. Geological Survey

Contents

List of Figures

Introduction

The *PhasePApy* has two sub-packages: the *PhasePicker* and *Associator*, aiming to identify phase arrival onsets and associate them to phase types respectively. The *PhasePicker* and *Associator* can work jointly or separately. Three autopickers are implemented in the *PhasePicker* sub-package: the frequency band picker (*FBpicker*), the Akaike Information Criteria (AIC) function derivative picker (*AICDpicker*), and the kurtosis picker (*KTpicker*). The *FBpicker* is a modified method from *Lomax et al.* (2012). The *FBpicker* automatically adapts to instrumentation sampling rate and no need to specify triggering threshold for P- or S-waves. The *AICDpicker* uses the derivation of the AIC function to determine the characteristic function (*Sleeman and van Eck*, 1999; *Maeda*, 1985). The *KTpicker* in the *PhasePicker* is a kurtosis-based picker (*Saragiotis et al.*, 2002; *Panagiotakis et al.*, 2008). There are two associators included in the *Associator* sub-package: the *1D* and *3D Associator*, which determine phase types for picks can best fit potential earthquakes by minimizing root mean square (RMS) residuals of the misfits. Both two associators use travel-time look up tables to determine the best estimation of the earthquake location and evaluate the phase type for picks.

This program is developed to leverage the growing number of scientific libraries being written in *Python* mostly notably *Obspy* (*Beyreuther et al.*, 2010). The choice of making our algorithms compatible with *Obspy* means that all data formats and access methods supported by *Obspy* are naturally supported, such as SEED, MiniSEED, SAC, SEGY, and etc. The other libraries on which *Obspy* and our algorithms rely are invaluable *Numpy* (*Oliphant*, 2007), *Scipy* (*Jones et al.*, 2001), and *Matplotlib* (*Hunter*, 2007).

The *PhasePApy* uses those open libraries to implement the picker routines. The

PhasePicker requires the *Python* libraries *Obspy*, *Numpy*, and *Scipy*, maybe *Matplotlib* if the user need visualize the performance. The *Associator* requires several open libraries and modules for the processing such as *Numpy*, *Sqlalchemy*, *datetime*, *operator*, *itertools*, and *search*, maybe more modules for plotting (check plot modules in *Associator* sub-package if necessary). This documentation provides two examples to demonstrate how to tune the parameters in these two packages. The code is colored in green, output in blue, and parameter description in red.

PhasePicker

FBpicker

```
#!/usr/bin/env python
from PhasePApy.PhasePicker import fbpicker
from obspy.core import read
#================================================================
# FBpicker example
file = open('../../20130616153750/OKCFA.HHZ.OK...20130616153750.msd')
st = read(file); tr = st[0]; tr.detrend('linear')
chenPicker = fbpicker.FBPicker (t_long = 5, freqmin = 1, mode = 'rms', t_ma = 20,
nsigma = 6, t_up = 0.4, nr_len = 2, nr_coeff = 2, pol_len = 10, pol_coeff = 10, un-
cert_coeff = 3)
scnl, picks, polarity, snr, uncert = chenPicker.picks(tr)
print 'scnl:', scnl
print 'picks:', picks
print 'polarity:', polarity
print 'signal to noise ratio:', snr
print 'uncertainty:', uncert
# plot
summary = fbpicker.FBSummary(chenPicker, tr)
summary.plot_bandfilter()
summary.plot_statistics()
summary.plot_summary()
```

Output:

scnl: OKCFA.HHZ.OK.

picks: [UTCDateTime(2013, 6, 16, 15, 38, 56, 310000), UTCDateTime(2013, 6, 16, 15, 42, 11, 550000)]

polarity: ['D', ' ']

signal to noise ratio: [31.2 8.7]

uncertainty: [0.03 0.02]

Line 1: set up *Python* environment if there are several versions of *Python* installed.

Line 2 and 3: import *PhasePicker* and required libraries.

Line 4: create the file handle of the data.

Line 5: read in the steam, trace, and remove the trend of the trace, we recommend the removal of the trend when choose the rms mode.

Line 6: create the instance of the *FBpicker* such as *chenPicker*.

Parameter description:

t_long: the time in seconds of moving window to calculate CF_n of each band pass filtered data

freq_min: the center frequency of first octave filtering band

cnr: corner order of band pass filtering

t_ma: the time in seconds of the moving average window for dynamic threshold

n_sigma: controls the level of threshold to trigger potential picks

t_up: the time in seconds not allowed consecutive pick in this duration

mode: two options: standard deviation (std) or root mean square (rms)

nr_len: noise ratio filter window length before and after potential picks used to calculate standard deviation

nr_coeff: control threshold level to determine if remove the pick by comparing std

or rms on both sides of each potential pick

pol_len: window length in samples to calculate the standard deviation of waveform before the picks

pol_coeff: determine if declare first motion as 'Compression' or 'Dilation' by comparing the first local extreme value after pick and standard deviation in previous window

uncert_len: window length in time to calculate the rms of the *CF* before the picks, we make it as long as t_ma

uncert_coeff: control the floating level based on the noise of *CF*

Line 7: pass in the trace to *chenPicker* to obtain results.

Line 8–12: print out the processing results.

Line 13–16: plot filtered data BF_n (Fig. 1), CF_n (Fig. 2), and CF (Fig. 3).

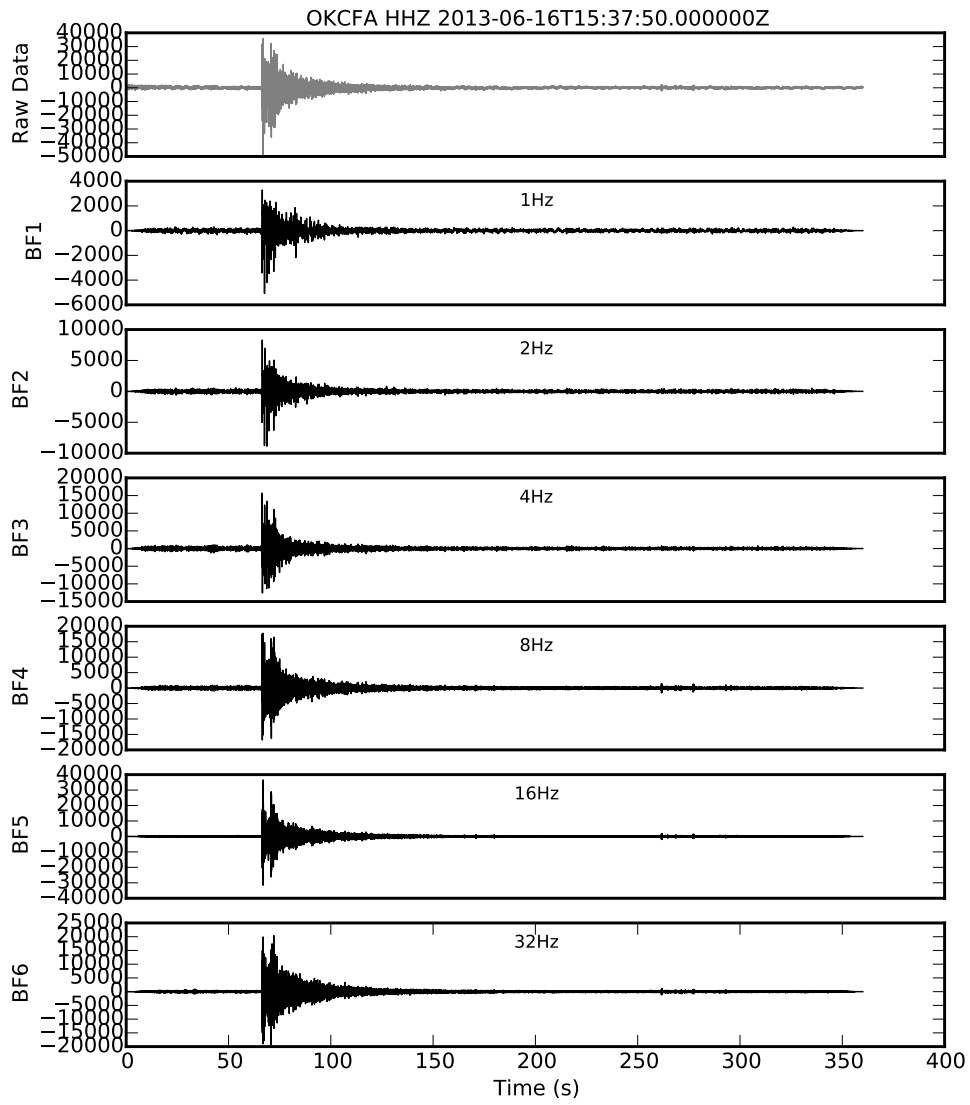


Figure 1: A de-trended example seismogram (top panel) from vertical component of station OKCFA in Oklahoma regional network (OK) is octave-filtered with 6 determined bands (BF_1 - BF_6) from the *FBpicker*. The sampling rate of the data is 100 Hz. The central frequency of each band is labeled in the middle of each panel. A 10% tapering is applied to all bands.

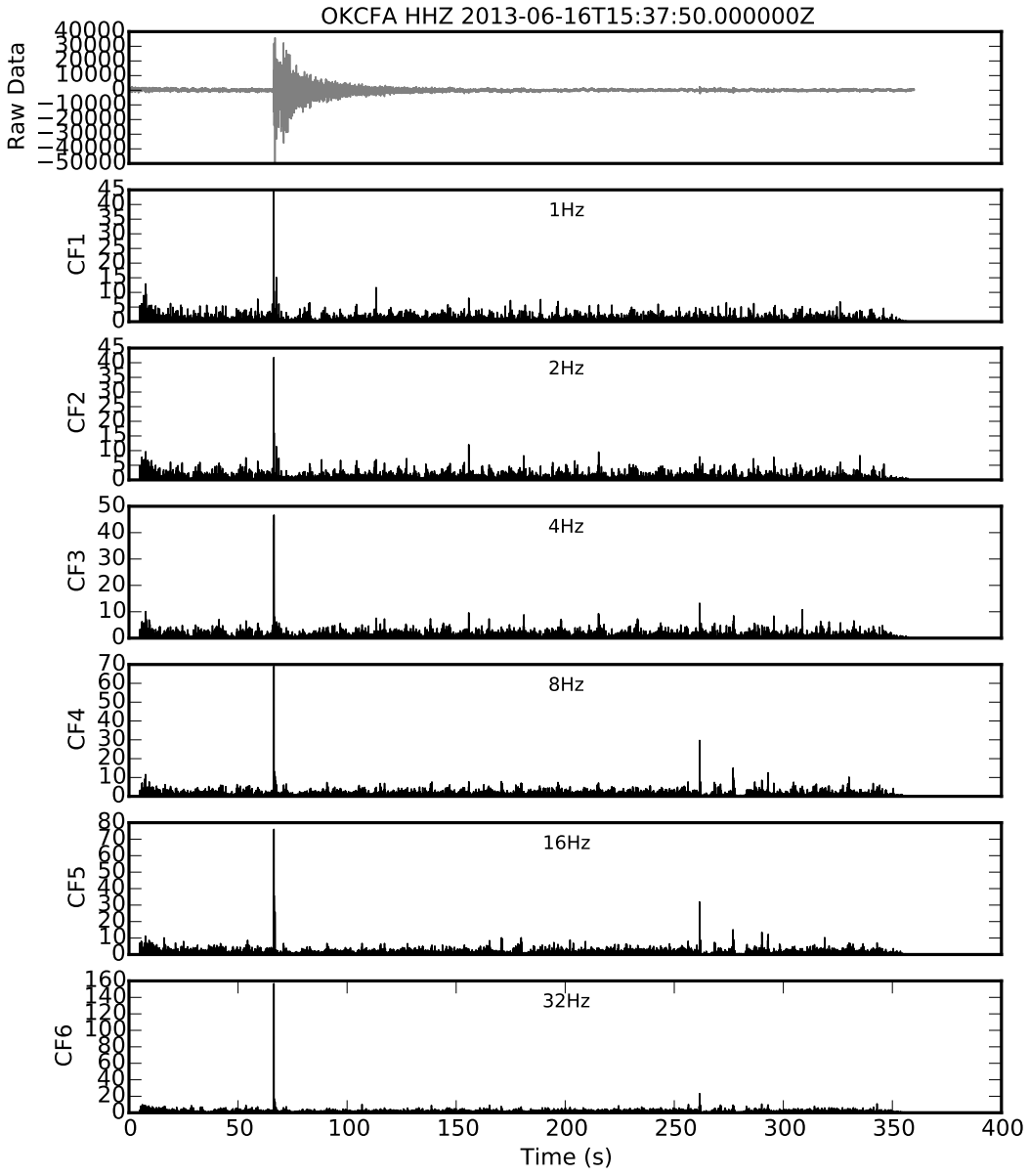


Figure 2: The de-trended seismogram on vertical channel from OKCFA (top panel) and the *FBpicker* determined the $CF_n(CF_1$ - $CF_6)$ with RMS mode.

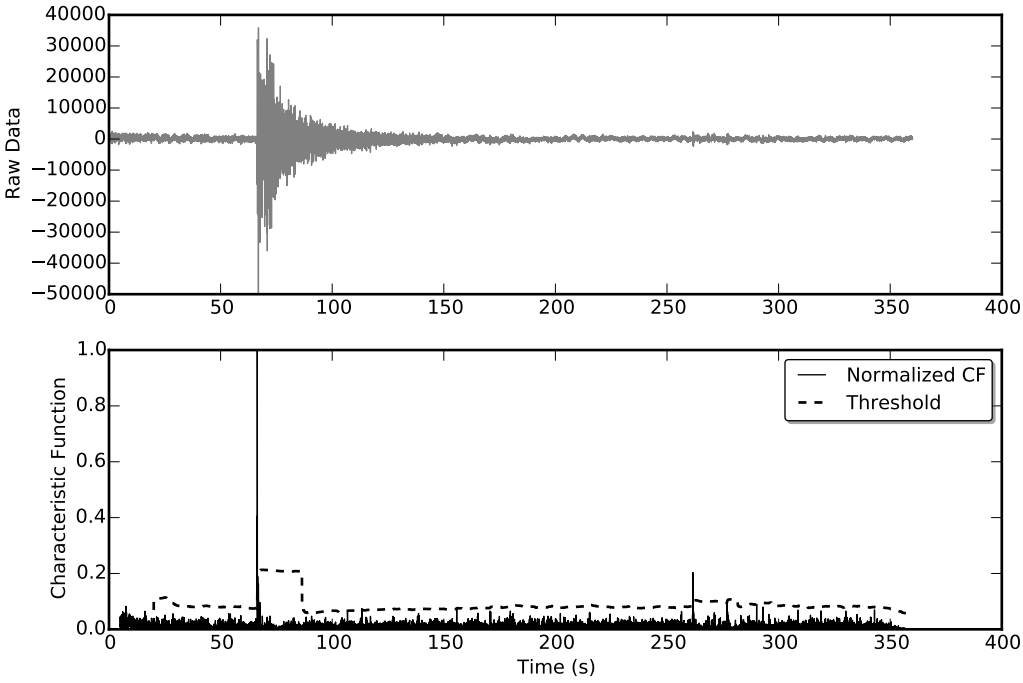


Figure 3: The upper panel is the de-trended seismogram on vertical channel from OKCFA. The *FBpicker* summarized the *CF* of all bands of is normalized in lower panel (black solid line), and the dynamic threshold level of 6 is the dashed line.

AICDpicker

```
#!/usr/bin/env python
from PhasePApy.PhasePicker import aicdpicker
from obspy.core import read
#========================================================
# AICDpicker example
file = open('../../20130616153750/OKCFA.HHZ.OK...20130616153750.msd')
st = read(file); tr = st[0]; tr.detrend('linear')
chenPicker = aicdpicker.AICDPicker (t_ma = 3, nsigma = 6, t_up = 0.78, nr_len = 2,
nr_coeff = 2, pol_len = 10, pol_coeff = 10, uncert_coeff = 3)
scnl, picks, polarity, snr, uncert = chenPicker.picks(tr)
print 'scnl:', scnl
print 'picks:', picks
print 'polarity:', polarity
print 'signal to noise ratio:', snr
print 'uncertainty:', uncert
# plot
summary = aicdpicker.AICDSummary(chenPicker, tr)
summary.plot_summary()
```

Output:

scnl: OKCFA.HHZ.OK.

picks: [UTCDateTime(2013, 6, 16, 15, 38, 56, 330000)]

polarity: ['D']

signal to noise ratio: [237.7]

uncertainty: [0.01]

Line 1: set up *Python* environment if there are several versions of *Python* installed.

Line 2 and 3: import *PhasePicker* and required libraries.

Line 4: create the file handle of the data.

Line 5: read in the steam, trace, and remove the trend of the trace, we recommend the removal of the trend when choose the rms mode.

Line 6: create the instance of the *AICDpicker* such as *chenPicker*.

Parameter description:

t_ma: the time in seconds of the moving average window for dynamic threshold

n_sigma: controls the level of threshold to trigger potential picks

t_up: the time in seconds not allowed consecutive pick in this duration

nr_len: noise ratio filter window length before and after potential picks used to calculate standard deviation

nr_coeff: control threshold level to determine if remove the pick by comparing std or rms on both sides of each potential pick

pol_len: window length in samples to calculate the standard deviation of waveform before the picks

pol_coeff: determine if declare first motion as 'Compression' or 'Dilation' by comparing the first local extreme value after pick and standard deviation in previous window

uncert_len: window length in time to calculate the rms of the *CF* before the picks, we make it as long as t_ma

uncert_coeff: control the floating level based on the noise of *CF*

Line 7: pass in the trace to *chenPicker* to obtain results.

Line 8–12: print out the processing results.

Line 13 and14: plot *CF* (Fig. 4).

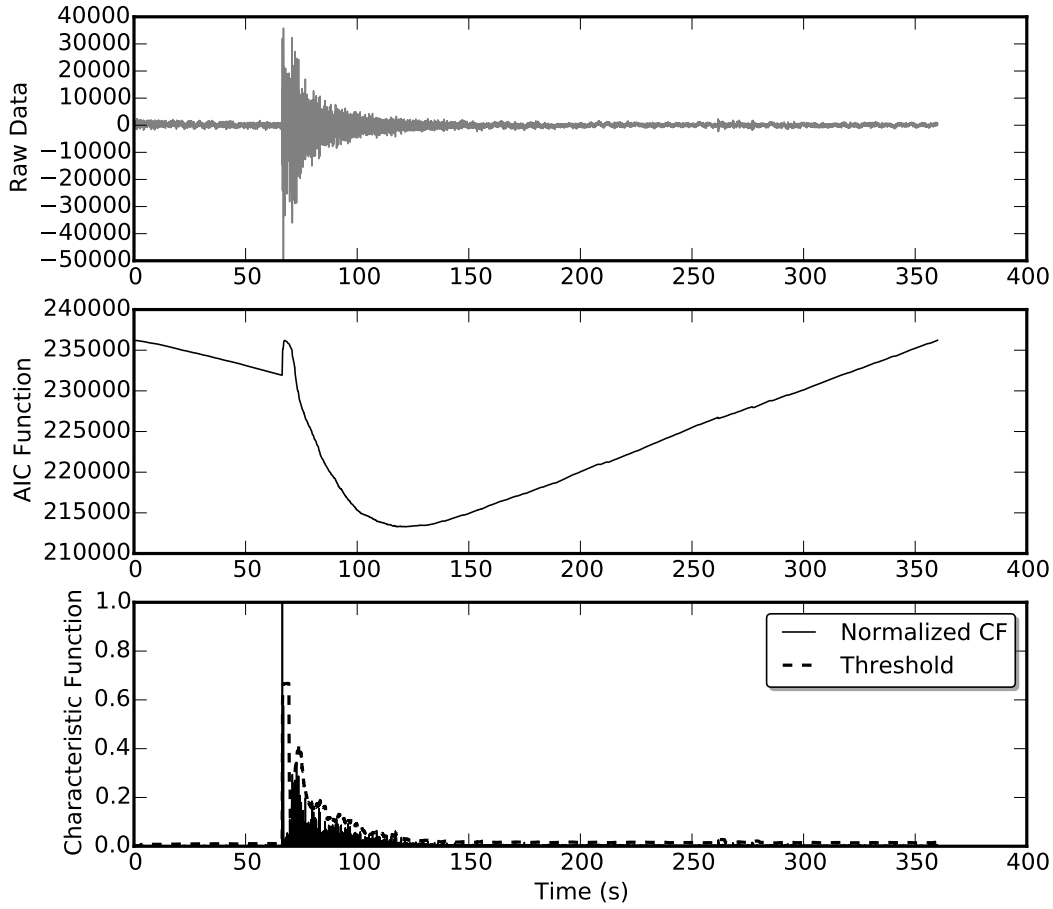


Figure 4: The upper panel is the de-trended raw seismogram on vertical channel from OKCFA; the middle panel is the AIC function of data; the lower panel is the normalized derivative of the AIC function, and the dynamic threshold level of 6 is indicated in dashed line.

KTpicker

```
#!/usr/bin/env python
from PhasePApy.PhasePicker import ktpicker
from obspy.core import read
#========================================================
# KTpicker example
file = open('../../20130616153750/OKCFA.HHZ.OK...20130616153750.msd')
st = read(file); tr = st[0]; tr.detrend('linear')
chenPicker = ktpicker.KTPicker (t_win = 1, t_ma = 10, nsigma = 6, t_up = 0.78, nr_len
= 2, nr_coeff = 2, pol_len = 10, pol_coeff = 10, uncert_coeff = 3)
scnl, picks, polarity, snr, uncert = chenPicker.picks(tr)
print 'scnl:', scnl
print 'picks:', picks
print 'polarity:', polarity
print 'signal to noise ratio:', snr
print 'uncertainty:', uncert
# plot
summary = ktpicker.KTSummary(chenPicker, tr)
summary.plot_summary()
```

Output:

scnl: OKCFA.HHZ.OK.

picks: [UTCDateTime(2013, 6, 16, 15, 38, 56, 340000), UTCDateTime(2013, 6, 16, 15, 42, 11, 570000)]

polarity: ['D', 'C']

signal to noise ratio: [51.1 12.3]

uncertainty: [0.03 0.01]

Line 1: set up *Python* environment if there are several versions of *Python* installed.

Line 2 and 3: import *PhasePicker* and required libraries.

Line 4: create the file handle of the data.

Line 5: read in the steam, trace, and remove the trend of the trace, we recommend the removal of the trend when choose the rms mode.

Line 6: create the instance of the *KTpicker* such as *chenPicker*.

Parameter description:

t_win: the time in seconds of moving window to calculate kurtosis

t_ma: the time in seconds of the moving average window for dynamic threshold

n_sigma: controls the level of threshold to trigger potential picks

t_up: the time in seconds not allowed consecutive pick in this duration

nr_len: noise ratio filter window length before and after potential picks used to calculate standard deviation

nr_coeff: control threshold level to determine if remove the pick by comparing std or rms on both sides of each potential pick

pol_len: window length in samples to calculate the standard deviation of waveform before the picks

pol_coeff: determine if declare first motion as 'Compression' or 'Dilation' by com-

paring the first local extreme value after pick and standard deviation in previous window

uncert_len: window length in time to calculate the rms of the *CF* before the picks, we make it as long as t_ma

uncert_coeff: control the floating level based on the noise of *CF*

Line 7: pass in the trace to *chenPicker* to obtain results.

Line 8–12: print out the processing results.

Line 13 and14: plot *CF* (Fig. 5).

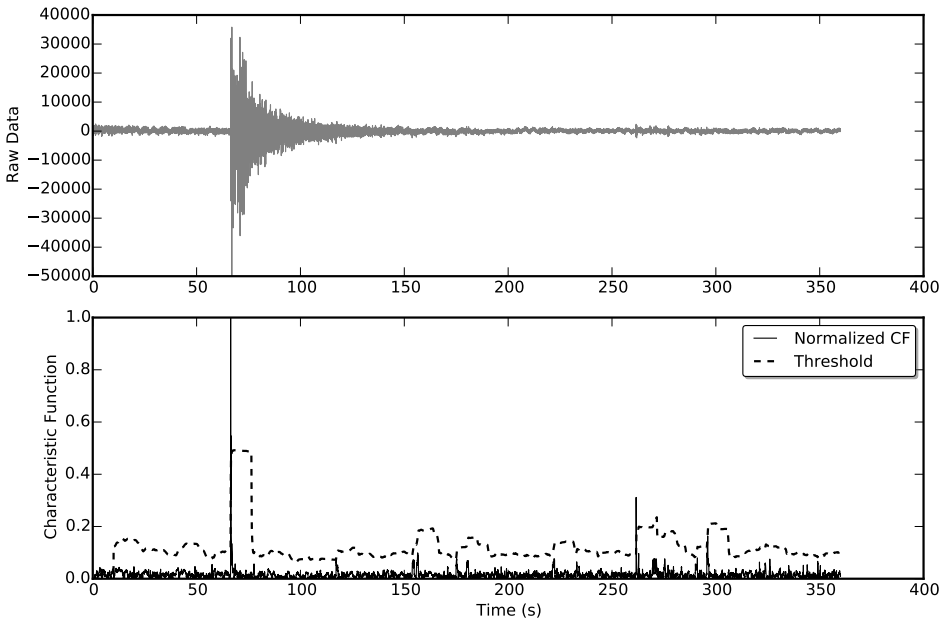


Figure 5: The de-trended vertical channel seismogram from OKCFA (upper panel). The *KTpicker* normalized *CF* is in lower panel, and the dynamic threshold level of 6 is indicated by the dashed line.

Associator

1D Associator

```
#!/usr/bin/env python
from PhasePApy.PhasePicker import *
from obspy.core import read
import glob
import time
from sqlalchemy.orm import *
from sqlalchemy import create_engine
from PhasePApy.Associator import tables1D, assoc1D, plot1D
from datetime import datetime
# databases
db_assoc = 'sqlite:///associator1D.db'
db_tt = 'sqlite:///tt_stations_1D.db'
# connecting
engine_assoc = create_engine('db_assoc', echo = False)
tables1D.Base.metadata.create_all(engine_assoc)
Session = sessionmaker(bind = engine_assoc)
session = Session()
# glob data
files = glob.glob('../../20130616153750/*.msd')
for file in files:
    st = read(file)
    for i in range(len(st)):
        tr = st[i]; #print tr.stats
```

```
        # FBpicker example
        tr.detrend('linear')
        now = time.time()
        chenPicker = FBpicker(t_long = 5, freqmin = 1, mode = 'rms', t_ma =
        20, nsigma = 6, t_up = 0.78, nr_len = 2, nr_coeff = 2, pol_len = 10,
        pol_coeff = 10, uncert_coeff = 3)
        scnl, picks, polarity, snr, uncert = chenPicker.picks(tr)
        print "It took %f s to pick." %(time.time()-now)
        # populate database with picks
        if not picks: continue
        t_create = datetime.utcnow()
        for i in range(len(picks)):
        pick = picks[i].datetime
        new_line = tables1D.Pick(scnl, pick, polarity[i], snr[i], uncert[i], t_create)
        session.add(new_line)
        session.commit()
# associate picks with phase types
chensAssoc = assoc1D.LocalAssociator(db_assoc, db_tt, max_km = 350, aggregation
= 1, aggr_norm = 'L2', cutoff_outlier = 10, assoc_ot_uncert = 7, nsta_declare = 4,
loc_uncert_thresh = 0.2)
chensAssoc.id_candidate_events()
chensAssoc.associate_candidates()
chensAssoc.single_phase()
# Plot
plt = plot1D.Plot(db_assoc, db_tt)
plt.cluster_plot(assoc_ot_uncert = 3)
```

plt.event_plot(1)

plt.section_plot(1, files)

Output:

It took 1.006072s to pick.

It took 1.003096s to pick.

It took 1.013223s to pick.

It took 3.117803s to pick.

It took 1.570385s to pick.

It took 1.483543s to pick.

It took 1.018244s to pick.

It took 1.025424s to pick.

It took 1.019520s to pick.

It took 1.024057s to pick.

It took 1.008089s to pick.

It took 1.025361s to pick.

It took 0.135289s to pick.

It took 0.116681s to pick.

It took 0.111830s to pick.

It took 1.023925s to pick.

It took 1.007293s to pick.

It took 1.005245s to pick.

It took 0.537876s to pick.

It took 0.445709s to pick.

It took 0.536248s to pick.

It took 0.467710s to pick.

It took 0.507865s to pick.

It took 0.459343s to pick.

It took 1.021057s to pick.

It took 1.017043s to pick.

It took 1.018937s to pick.

event_id: 1

ot: 2013-06-16 15:38:50.483549 ot_uncert: 0.319 loc: 35.476 -97.091 loc_uncert: 0.03

nsta: 6

Line 1: set up *Python* environment if there are several versions of *Python* installed.

Line 2–9: import necessary packages and modules.

Line 10–11: index travel picks and travel times databases.

Line 12–15: access databases and create the session.

Line 16: glob data files.

Line 17–33: Take the *FBpicker* as the example to process all the data, store the picks into database, and output the processing time, and associated earthquake information.

Line 18: Create the instance of the *3D Associator*: chenAssoc.

Parameters:

- db_assoc: associator database
- db_tt: travel time table database
- max_km: maximum distance of S-P interval in distance
- aggregation: the coefficient multiplied to minimum travel time
- aggr_norm: L2: median; L1: mean
- assoc_ot_uncert: origin time uncertainty window
- nsta_declare: minimum station number to declare a earthquake

Line 19: Create the candidates of event.

Line 20: Associate picks to phase types.

Line 21: Associate single phase if possible.

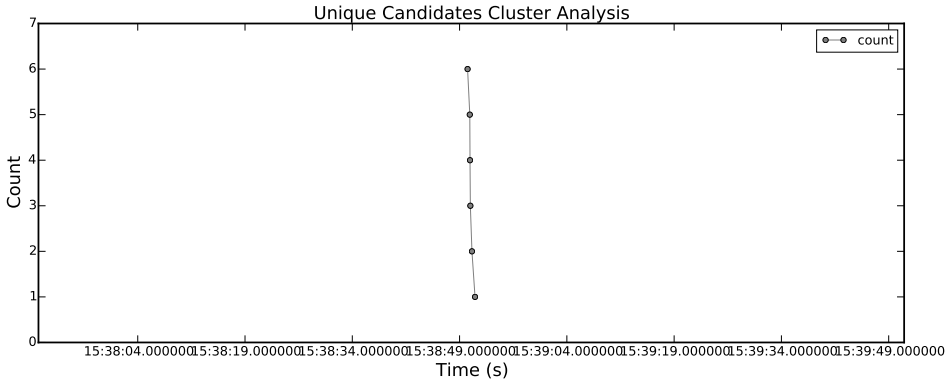
Line 22: Create plot instance.

Line 23: By tuning origin time uncertainty window, the user can choose proper window length in seconds for cluster analysis (Fig. 6). The window size of origin time uncertainty can affect the *Associator*'s performance if it is too small to include all the candidates close to the actual origin time.

Line 24: Plot event in map view (Fig. 7).

Line 25: Plot the phases in section view (Fig. 8).

(a)



(b)

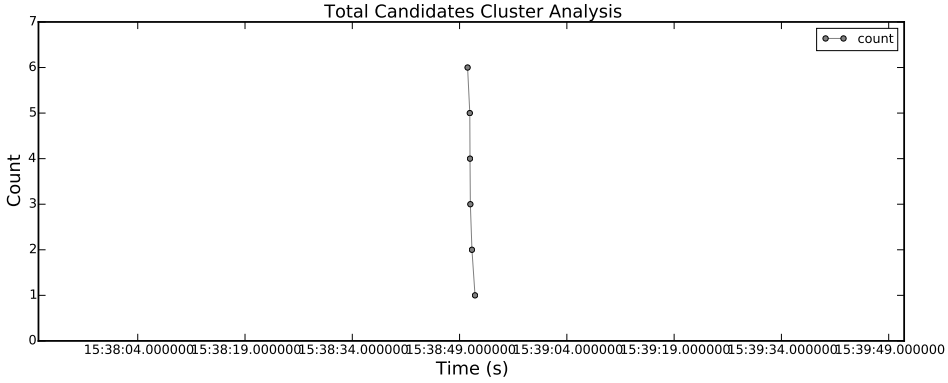


Figure 6: *1D Associator* cluster analysis of one-day data on June 16, 2013. (a) Cluster including concentric circles. (b) Cluster excluding concentric circles.

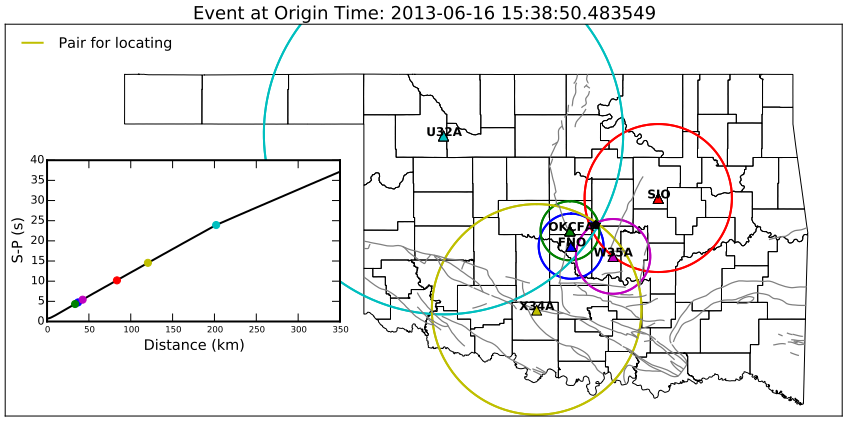


Figure 7: Map view of the event location determination with *1D Associator*. The triangles are the stations (cyan: U32A; blue: FNO; red: SIO; yellow: X34A; green: OKCFA; magenta: W35A). Circles are S-P interval in distance of event candidates. The star in the crossing area indicated the epicenter. The subplot shows the modeled S-P time curve and the *1D Associator*determined S-P times for event candidates.

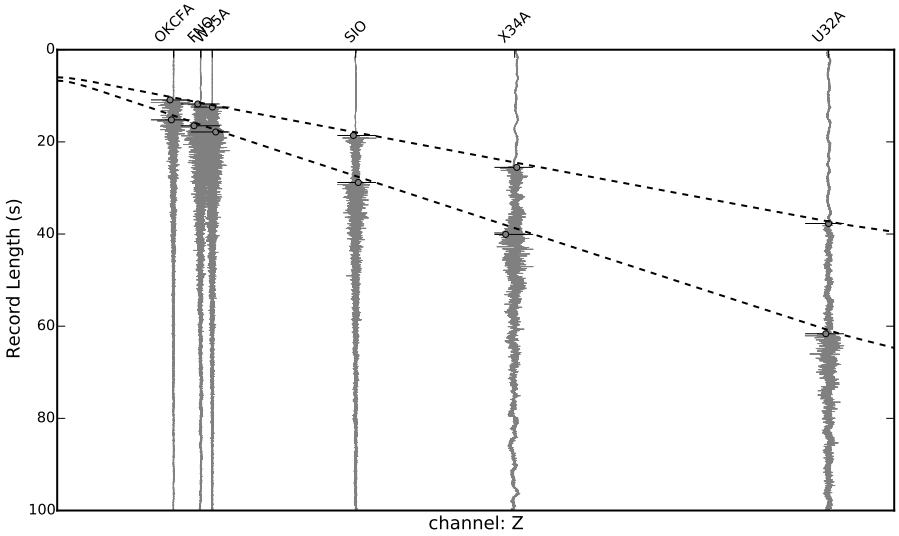


Figure 8: Cross section plot of the *FBpicker* and *1D Associator* performance on channel E. Two dash lines are phases travel time curves (Upper: P; Lower: S). Short bars: the associated picks from the stations. Gray dots: where the bars cross waveforms.

3D Associator

```
#!/usr/bin/env python
from PhasePApy.PhasePicker import *
from obspy.core import read
import glob
import time
from sqlalchemy.orm import *
from sqlalchemy import create_engine
from PhasePApy.Associator import tables3D, assoc3D, plot3D
from datetime import datetime
# databases
db_assoc = 'sqlite:///associator3D.db'
db_tt = 'sqlite:///tt_stations_3D.db'
# connecting
engine_assoc = create_engine('db_assoc ', echo = False)
tables3D.Base.metadata.create_all(engine_assoc)
Session = sessionmaker(bind = engine_assoc)
session = Session()
# glob data
files = glob.glob('../../20130616153750/*.msd')
for file in files:
    st = read(file)
    for i in range(len(st)):
        tr = st[i]; #print tr.stats
        # FBpicker example
```

```
            tr.detrend('linear')
            now = time.time()
            chenPicker = FBpicker(t_long = 5, freqmin = 1, mode = 'rms', t_ma =
            20, nsigma = 6, t_up = 0.78, nr_len = 2, nr_coeff = 2, pol_len = 10,
            pol_coeff = 10, uncert_coeff = 3)
            scnl, picks, polarity, snr, uncert = chenPicker.picks(tr)
            print "It took %f s to pick." %(time.time()-now)
            # populate database with picks
            if not picks: continue
            t_create = datetime.utcnow()
            for i in range(len(picks)):
            pick = picks[i].datetime
            new_line = tables3D.Pick(scnl, pick, polarity[i], snr[i], uncert[i], t_create)
            session.add(new_line)
            session.commit()
# associate picks with phase types
chensAssoc = assoc3D.LocalAssociator(db_assoc, db_tt, max_km = 350, aggregation
= 1, aggr_norm = 'L2', assoc_ot_uncert = 3, nsta_declare = 3, nt = 31, np = 41, nr =
5)
chensAssoc.id_candidate_events()
chensAssoc.associate_candidates()
# Plot
plt = plot3D.Plot(db_assoc, db_tt)
plt.cluster_plot(assoc_ot_uncert = 3)
plt.event_plot(1)
plt.section_plot(1, files)
```

Output:

It took 1.439758s to pick.

It took 1.039839s to pick.

It took 1.053400s to pick.

It took 3.151263s to pick.

It took 1.617033s to pick.

It took 1.491887s to pick.

It took 1.002975s to pick.

It took 0.996323s to pick.

It took 0.985801s to pick.

It took 1.004821s to pick.

It took 1.009234s to pick.

It took 1.026747s to pick.

It took 0.139633s to pick.

It took 0.113183s to pick.

It took 0.109697s to pick.

It took 1.031661s to pick.

It took 1.027996s to pick.

It took 1.030501s to pick.

It took 0.520301s to pick.

It took 0.452110s to pick.

It took 0.544002s to pick.

It took 0.468110s to pick.

It took 0.533928s to pick.

It took 0.459713s to pick.

It took 1.046138s to pick.

It took 1.042615s to pick.

It took 1.011989s to pick.

event_id: 1

ot: 2013-06-16 15:38:50.488549 ot_uncert: 0.635 loc: 35.45 -97.05 rms 0.753 nsta: 6

Line 1: set up *Python* environment if there are several versions of *Python* installed.

Line 2–9: import necessary packages and modules.

Line 10–11: index travel picks and travel times databases.

Line 12–15: access databases and create the session.

Line 16: glob data files.

Line 17–33: Take the *FBpicker* as the example to process all the data, store the picks into database, and output the processing time, and associated earthquake information.

Line 18: Create the instance of the *3D Associator*: chenAssoc.

Parameters:

- db_assoc: associator database
- db_tt: travel time table database
- max_km: maximum distance of S-P interval in distance
- aggregation: the coefficient multiplied to minimum travel time
- aggr_norm: L2: median; L1: mean
- assoc_ot_uncert: origin time uncertainty window
- nsta_declare: minimum station number to declare a earthquake
- nt, np, nr: node geometry

Line 19: Create the candidates of event.

Line 20: Associate picks to phase types.

Line 21: Create plot instance.

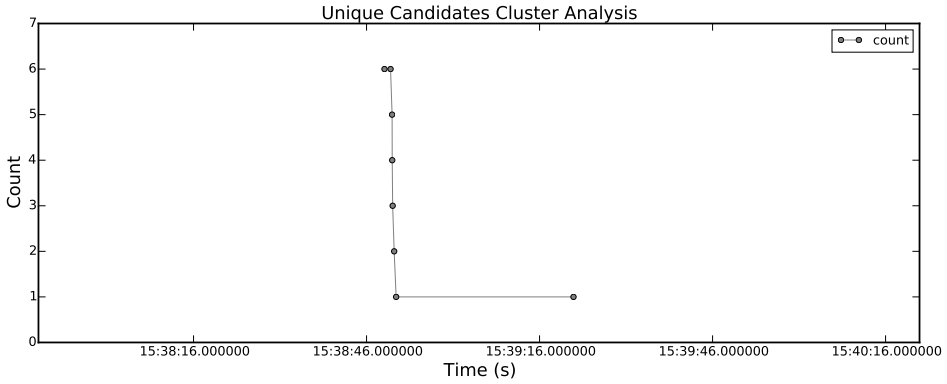
Line 22: By tuning origin time uncertainty window, the user can choose proper win-

dow length in seconds for cluster analysis (Fig. 9). The window size of origin time uncertainty can affect the *Associator*'s performance if it is too small to include all the candidates close to the actual origin time.

Line 23: Plot event in map view (Fig. 10).

Line 24: Plot the phases in section view (Fig. 11).

(a)



(b)

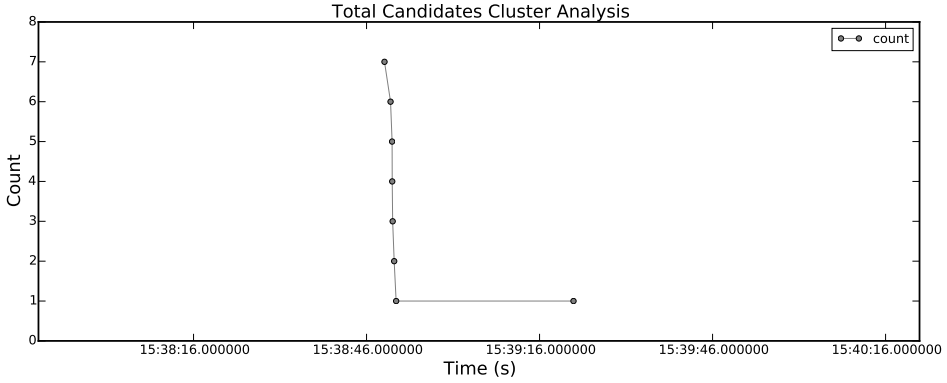


Figure 9: *3D Associator* cluster analysis of one-day data on June 16, 2013. (a) Cluster including concentric circles. (b) Cluster excluding concentric circles.

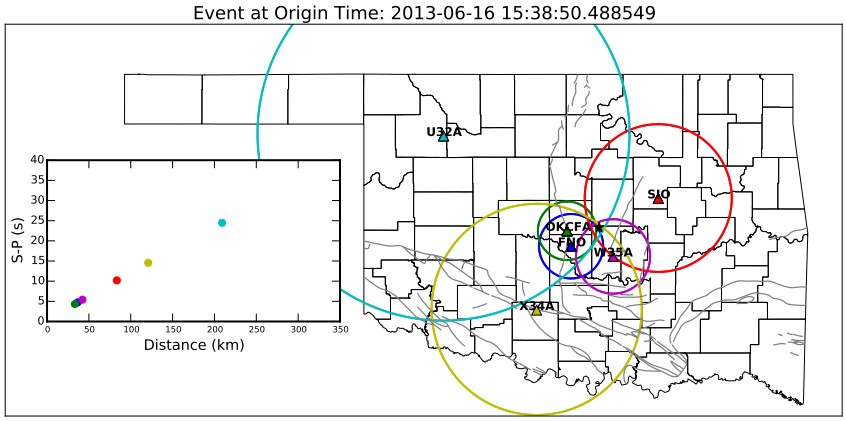


Figure 10: Map view of the event location determination with *3D Associator*. The triangles are the stations (cyan: U32A; blue: FNO; red: SIO; yellow: X34A; green: OKCFA; magenta: W35A). Circles are S-P interval in distance of event candidates. The star in the crossing area indicated the epicenter. The subplot shows the modeled S-P time curve and the *3D Associator*determined S-P times for event candidates.

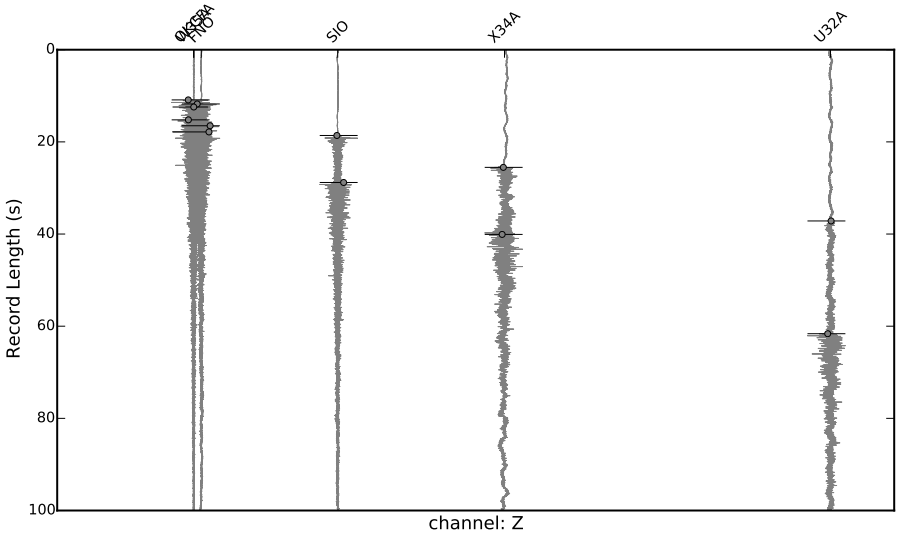


Figure 11: Cross section plot of the *FBpicker* and *3D Associator* performance on channel E. Two dash lines are phases travel time curves (Upper: P; Lower: S). Short bars: the associated picks from the stations. Gray dots: where the bars cross waveforms.

References

Beyreuther, M., R. Barsch, L. Krischer, T. Megies, Y. Behr, and J. Wassermann (2010), Obspy: A python toolbox for seismology, *Seismological Research Letter*, *81*(3), 530–533.

Hunter, J. D. (2007), Matplotlib: a 2d graphics evironment, *Computing in Science and Engineering*, *9*(3), 90–95.

Jones, E., E. Oliphant, and P. Peterson (2001), Scipy: Open source scientific tools for python, *http://www.scipy.org*.

Lomax, A., C. Satriano, and M. Vassallo (2012), Automatic picker developments and optimization: Filterpicker—a robust, broadband picker for real-time seismic monitoring and earthquake early warning, *Seismological Research Letter*, *83*(3), 531–540.

Maeda, N. (1985), A method for reading and checking phase times in autoprocessing system of seismic wave data, *Journal of the Seismological Society of Japan*, *38*(2), 365–379.

Oliphant, T. E. (2007), Python for scientific computing, *IEEE Compuing in Science and Enginerring*, *9*, 10–20.

Panagiotakis, C., E. Kokinou, and F. Vallianatos (2008), Automatic p-phase picking based on local-maxima distribution, *IEEE Transactions on Geoscience and Remote Sensing*, *46*(8), 2280–2287.

Saragiotis, C. D., L. J. Hadjileontiadis, and S. M. Panas (2002), Pai-s/k: A robust automatic seismic p phase arrival identification scheme, *IEEE Transactions on Geoscience and Remote Sensing*, *40*(6), 1395–1404.

Sleeman, R., and T. van Eck (1999), Robust automatic p-phase picking: an on-line implementation in the analysis of broadband seismogram recordings, *Physics of the Earth and Planetary Interiors*, *113*(1), 265–275.