Outline

1. Autokiwi - automatic data preparation
2. Cake - classical ray theory for Python
3. Snuffler - a waveform browser
4. Snufflings - plugins for Snuffler

Autokiwi

A Python script for automated event processing.

- download new event data
- apply preprocessing
- run a processing tool
- extract results
- push results to e.g. a web server

Autokiwi

Documentation with example configuration:

`http://kinherd.org/power/trac/wiki/AutokiwiTool`

How to set up an Autokiwi environment:

`http://kinherd.org/power/trac/wiki/`
`AutokiwiProcessingEnvironment`

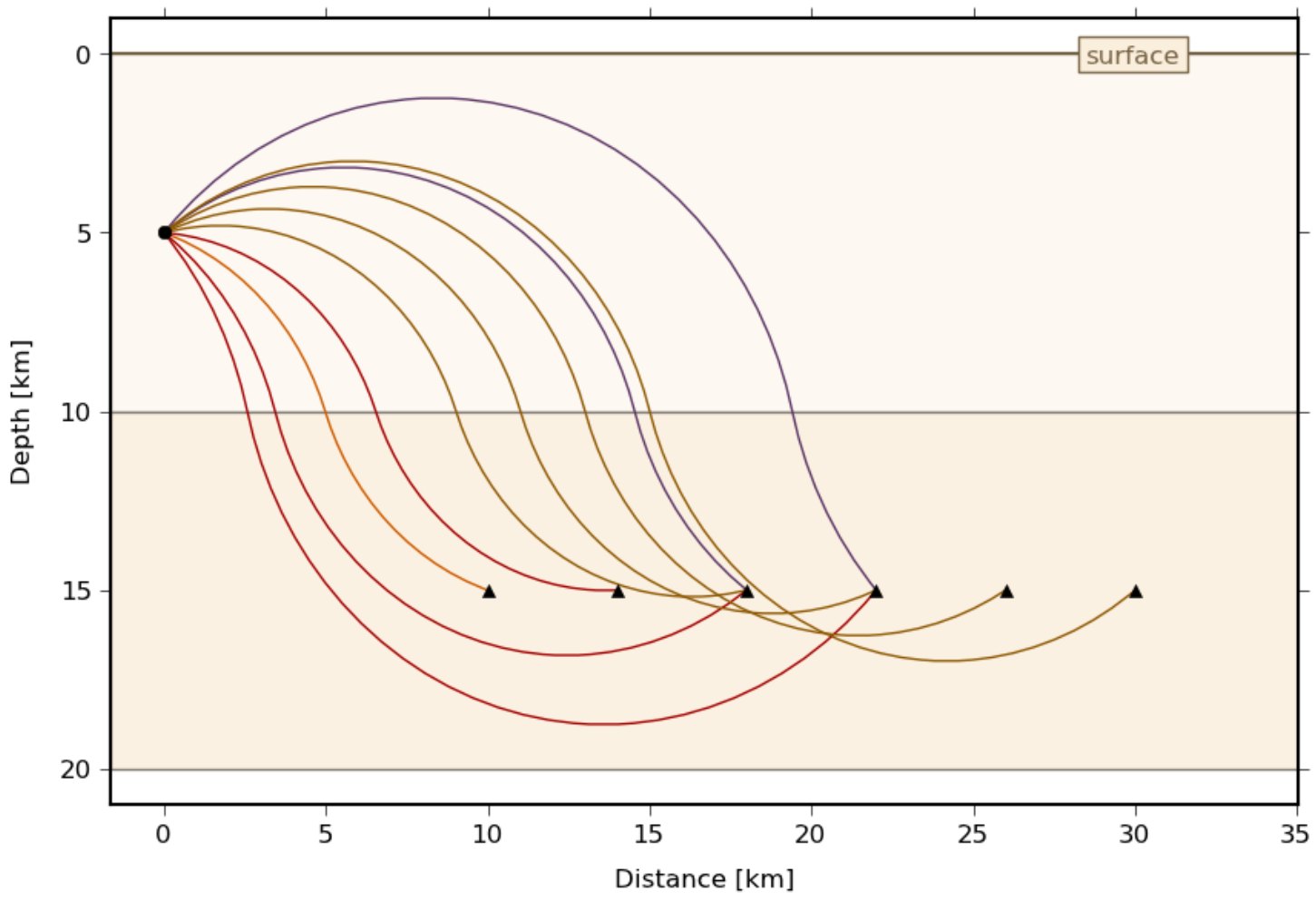
Cake

A tool to solve classical ray theory problems (for layer cake = 1D models).

- arrival times
- ray paths
- reflection and transmission coefficients
- geometrical spreading factors
- define any converted/reflected phase

Can be used as a standalone tool / as a Python module.

Cake




```
$ cake plot-rays --model=gradient.nd --sdepth=5 --rdepth=15 \
              --distances=10:30:6 --phases='P,p,P\,p\'
```

Cake

```
$ cake print --phase='Pv(moho)sPv10p'
Phase definition "Pv(moho)sPv10p":
 - P mode propagation, departing downward
 - upperside reflection with conversion from p to s at moho
 - S mode propagation, departing upward
 - surface reflection with conversion from s to p
 - P mode propagation, departing downward
 - upperside reflection at interface in 10 km depth
 - P mode propagation, departing upward
 - arriving at target from below
```

Cake: help and documentation

Command line tool help:

```
$ cake --help
```

Command line tool documentation:

`http://emolch.github.com/pyrocko/cake_doc.html`

Python module documentation:

`http://emolch.github.com/pyrocko/cake.html`

Snuffler: features

- can browse (large) data archives
- gapless processing
- manual picking
- display real time data
- extendable with Snufflings

Snuffler: documentation

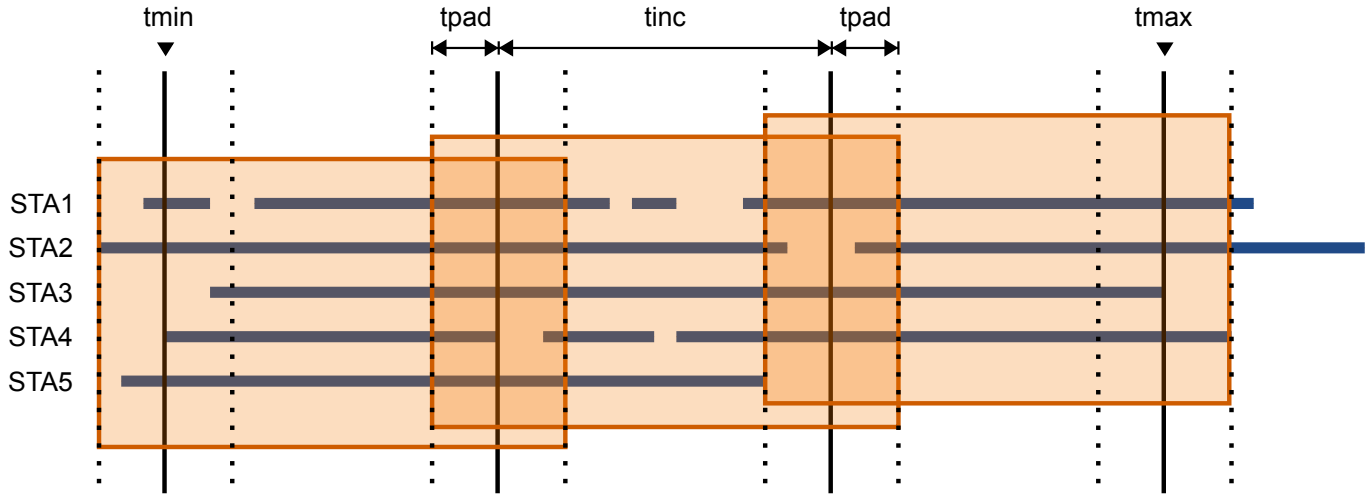
- 'snuffler –help'
- Help: press '?' in the viewer
- Manual:
  `http://emolch.github.com/pyrocko/snuffler.html`

Snuffler: working with markers

- There are three types of markers
  - normal markers
  - event markers - use ’e’ key
  - phase markers - use F-keys
- color markers with number keys ’0’-’5’
- write/read markers function is in the menu
- see ’help’ for more information

Snuffler: internals

Waveforms in Snuffler are accessed through a *Pile*
(class: pyrocko.pile.Pile)



A Pile is a powerful machine...

Why do we need a Pile?

A typical problem: downsample that dataset

- have a dataset (e.g. 12 stations, 8 months)
- mixed sampling rates (e.g. 100 Hz and 200 Hz)
- data gaps
- Mini-SEED files of approximately 1 hour each

Why we want a Pile

```
from pyrocko import pile, io, util
import time, calendar

# when pile.make_pile() is called without any arguments, the command line
# parameters given to the script are searched for waveform files and directories
p = pile.make_pile()

# get timestamp for full day before first data sample in all selected traces
tmin = util.day_start(p.tmin)

tinc = 3600.
tpad = 10.
target_deltat = 0.1

# iterate over the data, with a window length of one hour and 2x10 seconds of
# overlap
for traces in p.chopper(tmin=tmin, tinc=tinc, tpad=tpad):

    if traces: # the list could be empty due to gaps
        for tr in traces:
            tr.downsample_to(target_deltat, snap=True, demean=False)

            # remove overlapping
            tr.chop(tr.wmin, tr.wmax)

        window_start = traces[0].wmin
        timestring = util.time_to_str(window_start, format='%Y-%m-%d_%H')
        filepath = 'downsampled/%(station)s_%(channel)s_%(mytimestring)s.mseed'
        io.save(traces, filepath, additional={'mytimestring': timestring})
```

Snufflings

Snuffler is extendable with plugins: *Snufflings*
Example use cases:

- automatic picker / event detector
- add event markers from catalog
- cross correlate selected traces
- run location tool
- apply custom filter
- use external plotting tool
- acquire additional data
- ...

Snufflings: example

```
from pyrocko.snuffling import Snuffling
import numpy as num

class TestSnuffling(Snuffling):

    def setup(self):
        '''Customization of the snuffling.'''

        self.set_name('Integrate')
        self.set_live_update(False)


    def call(self):
        '''Main work routine of the snuffling.'''

        self.cleanup()

        for traces in self.chopper_selected_traces(fallback=True):
            for trace in traces:
                y = trace.get_ydata()
                y -= y.mean()
                trace.set_ydata(num.cumsum(y))
                trace.set_codes(location='integrated')
                self.add_trace(trace)

def __snufflings__():
   return [ TestSnuffling() ]
```

Exercise

Write a snuffling which plots particle motion. Use '$HOME/local/src/pyrocko/pyrocko/snufflings/ampspec.py' as a starting point.

Snuffling module documentation:

`http://emolch.github.com/pyrocko/snuffling.html`

More examples at
`http://emolch.github.com/pyrocko/snuffling.html`

and in

$HOME/local/src/pyrocko/pyrocko/snufflings