SAND2017-1961PE

BLOODHOUND

Sandia National Laboratories

A smart infrasound event detection system

POC: Stephen Arrowsmith, sjarrow@sandia.gov

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

U.S. DEPARTMENT OF ENERGY

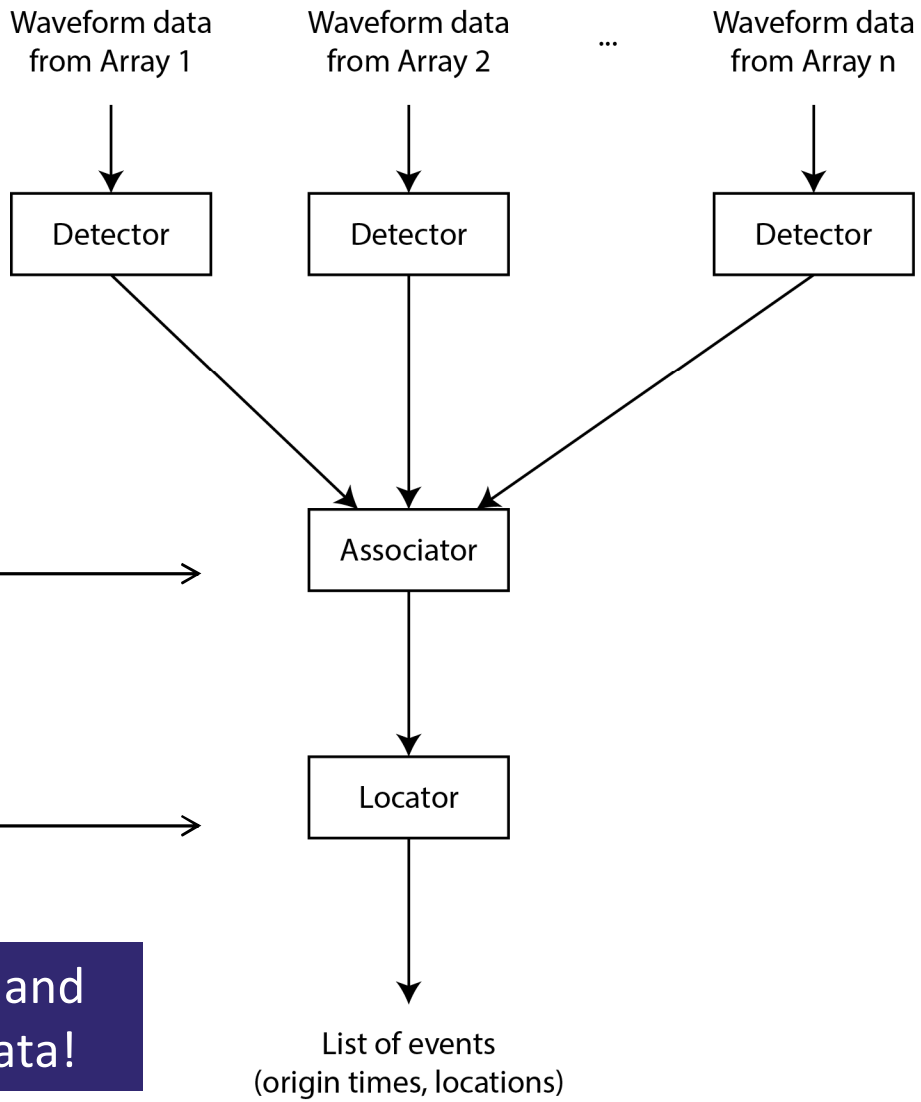
NNSA

The challenge of working with infrasound

Infrasound signals are complex and often buried in wind noise

The atmosphere is dynamic; infrasound propagates at slow speeds.

The atmosphere is dynamic



We can't take seismic-based methods and expect them to work for infrasound data!

Features

Core functionality

Automatic detection, association, location, and yield estimation of seismic and infrasound events

Smart detectors learn from data and fuse multiple features for targeted detection of signals of interest

(e.g., detection of moving vehicles, or detection of single point explosions at long range)

Network processing algorithms can exploit meteorological data to enhance event location.

In-house developed full-wave infrasound yield estimation tools provide better estimates than traditional parametric techniques for estimating explosion yields

Additional features

Infrasound propagation simulations

Built-in geometric propagation codes and interfaces to open-source community codes

Analyst tools for review

A wide variety of built-in tools for analyst review that are easily extensible in Python (various time series plots, PMCC plots, record sections, spectrograms, spectra, etc.)

Smart detectors

Bloodhound combines multiple detectors for more targeted detection of signals of interest

Why BLOODHOUND?

Existing detectors use only single feature (e.g., power) to separate signals and noise.

Bloodhound intelligently combines a parallel bank of detectors to provide more targeted detection of signals of interest (this fuses the detection and discrimination problems).

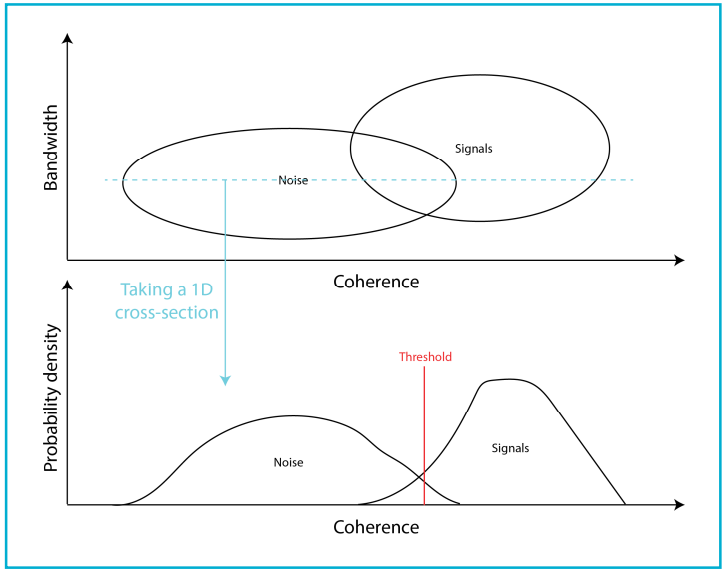
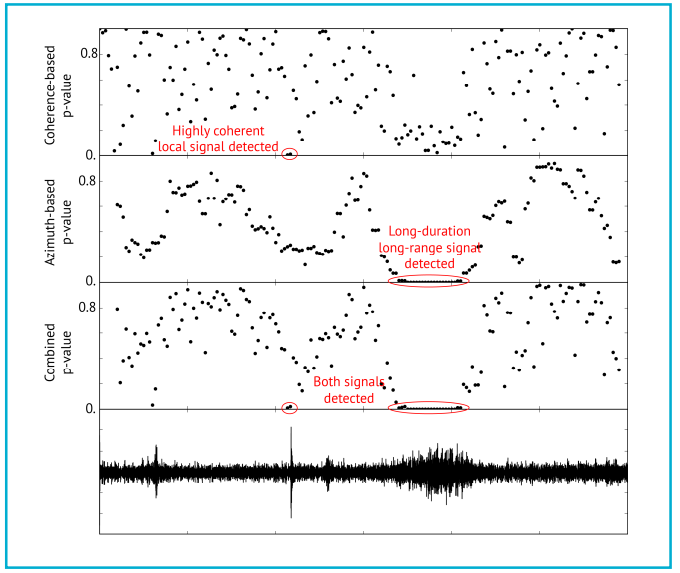


Illustration of a two-feature detector (top). Most existing detectors are developed for a single feature (bottom).



A multivariate detector based on two features detects multiple signals of interest and provides additional details of signals.

Generating automatic infrasound catalogs

Bloodhound contains unique algorithms for associating and locating 'smart' detections at multiple arrays

Why BLOODHOUND?

Existing tools require extensive manual work to analyze data.

Bloodhound is a pipeline tool that can be set up to chew a huge amount of data and generate on-the-fly catalogs. It uses multithreading so it runs fast on standard desktop computers.

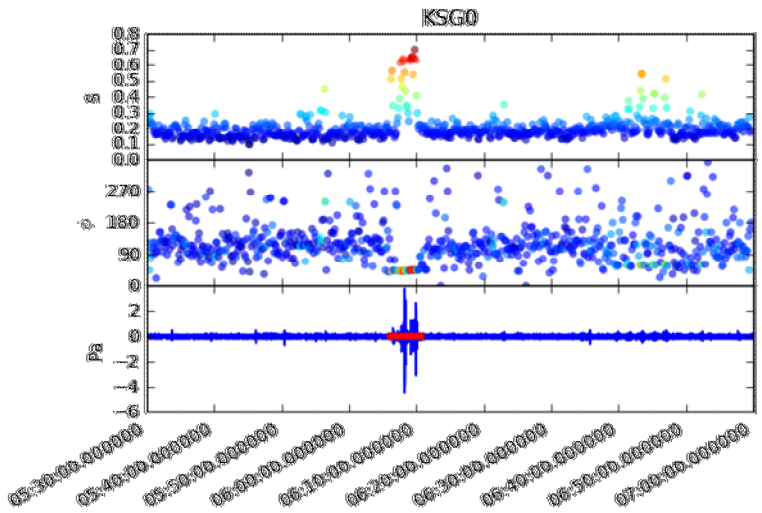
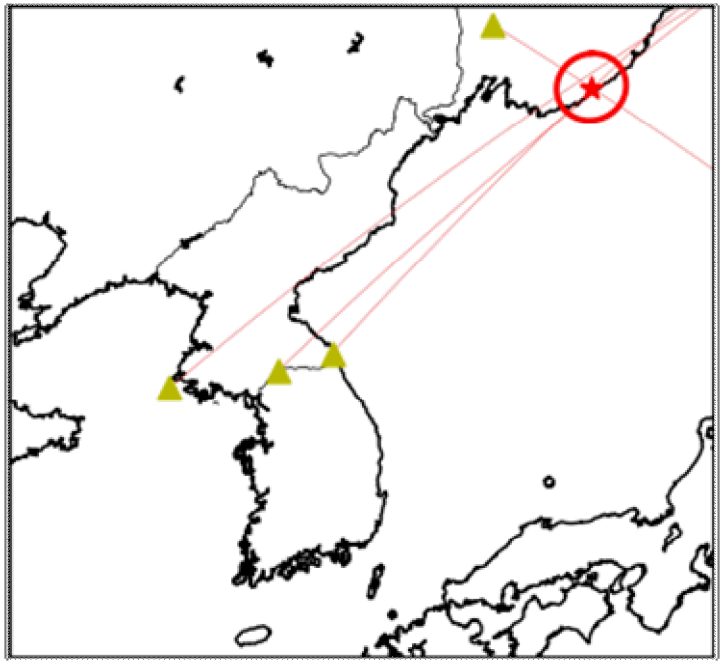
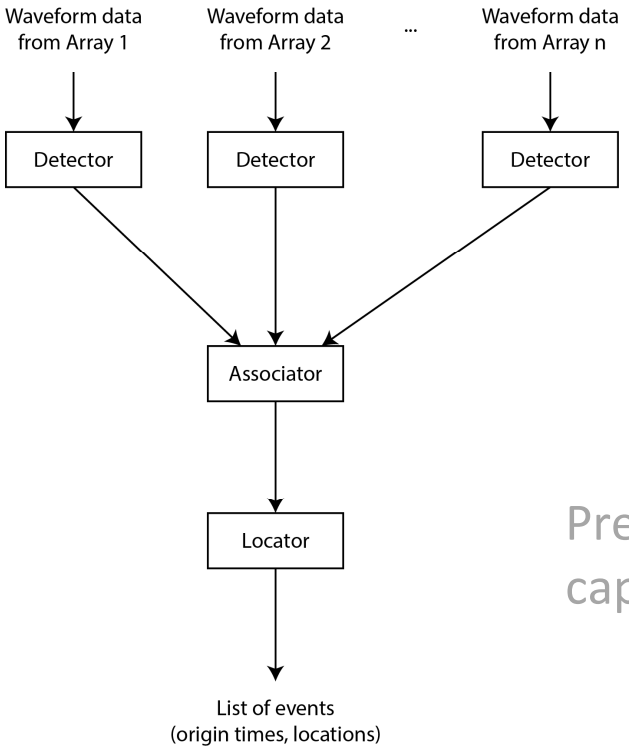


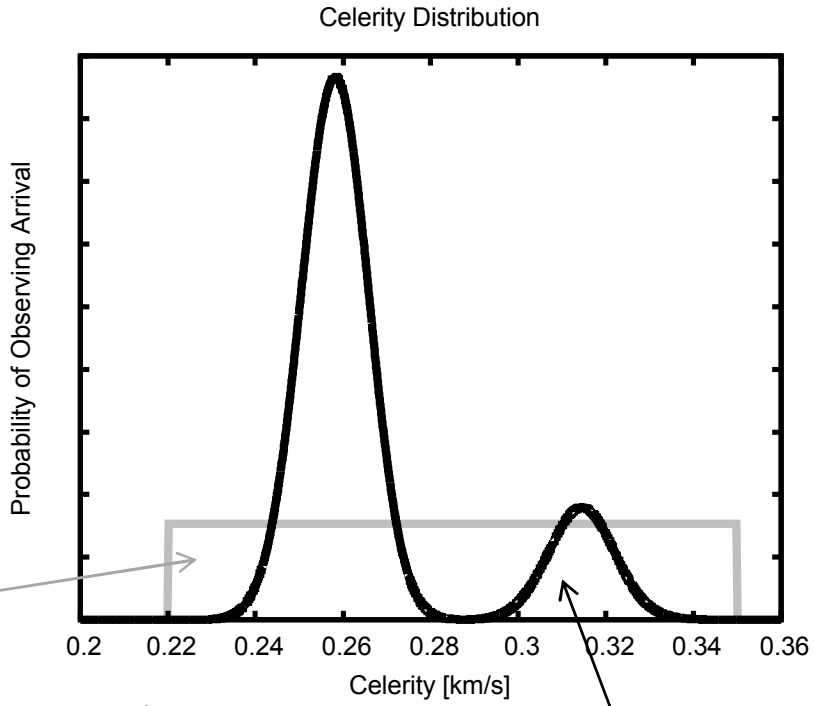
Illustration of an event detected in Russia by BLOODHOUND. The signal shown above is from KSG.



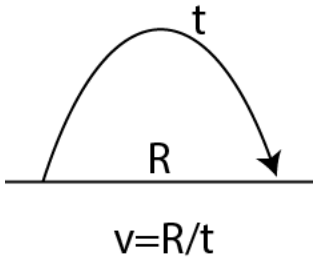
Generating automatic infrasound catalogs

Bloodhound features smarter associators





Previous capability



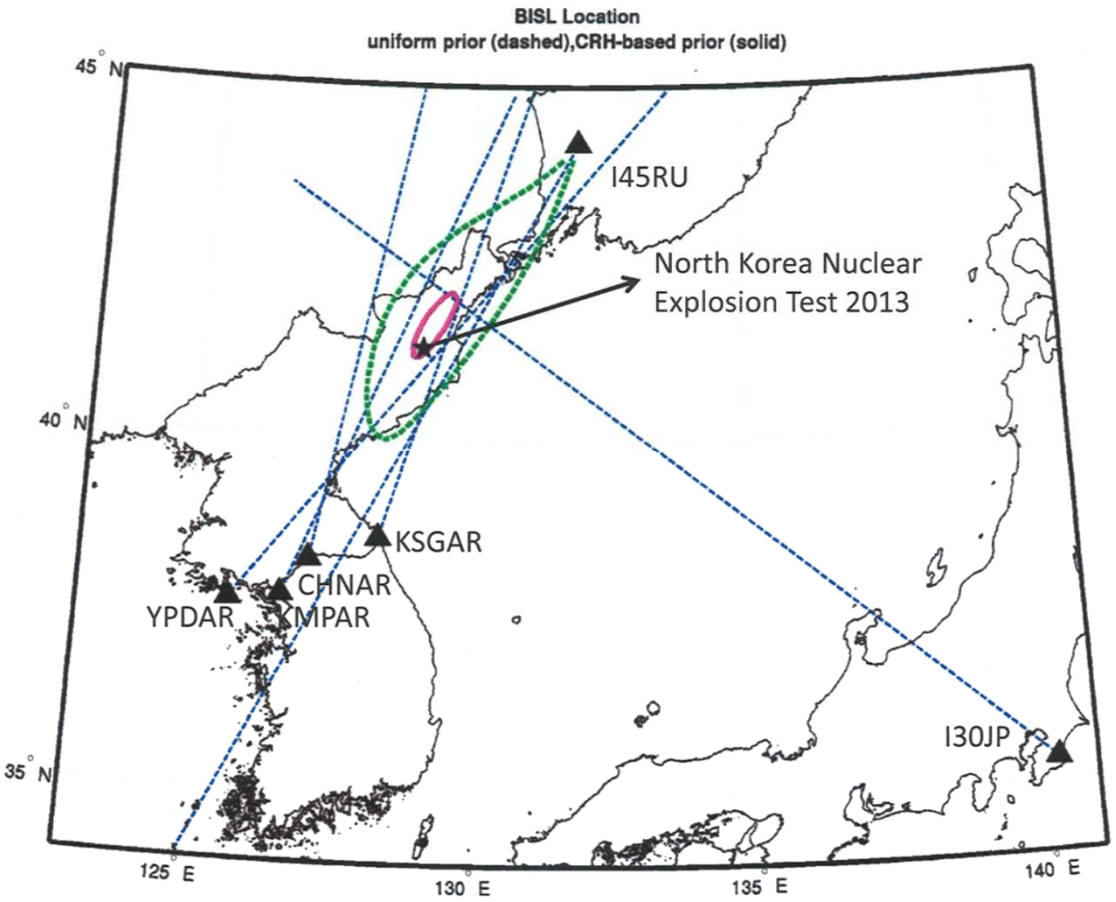
$v=R/t$

Celerity (v) is representative of the 'phase' or propagation path

We are building improved models that use atmospheric and propagation models

Generating automatic infrasound catalogs

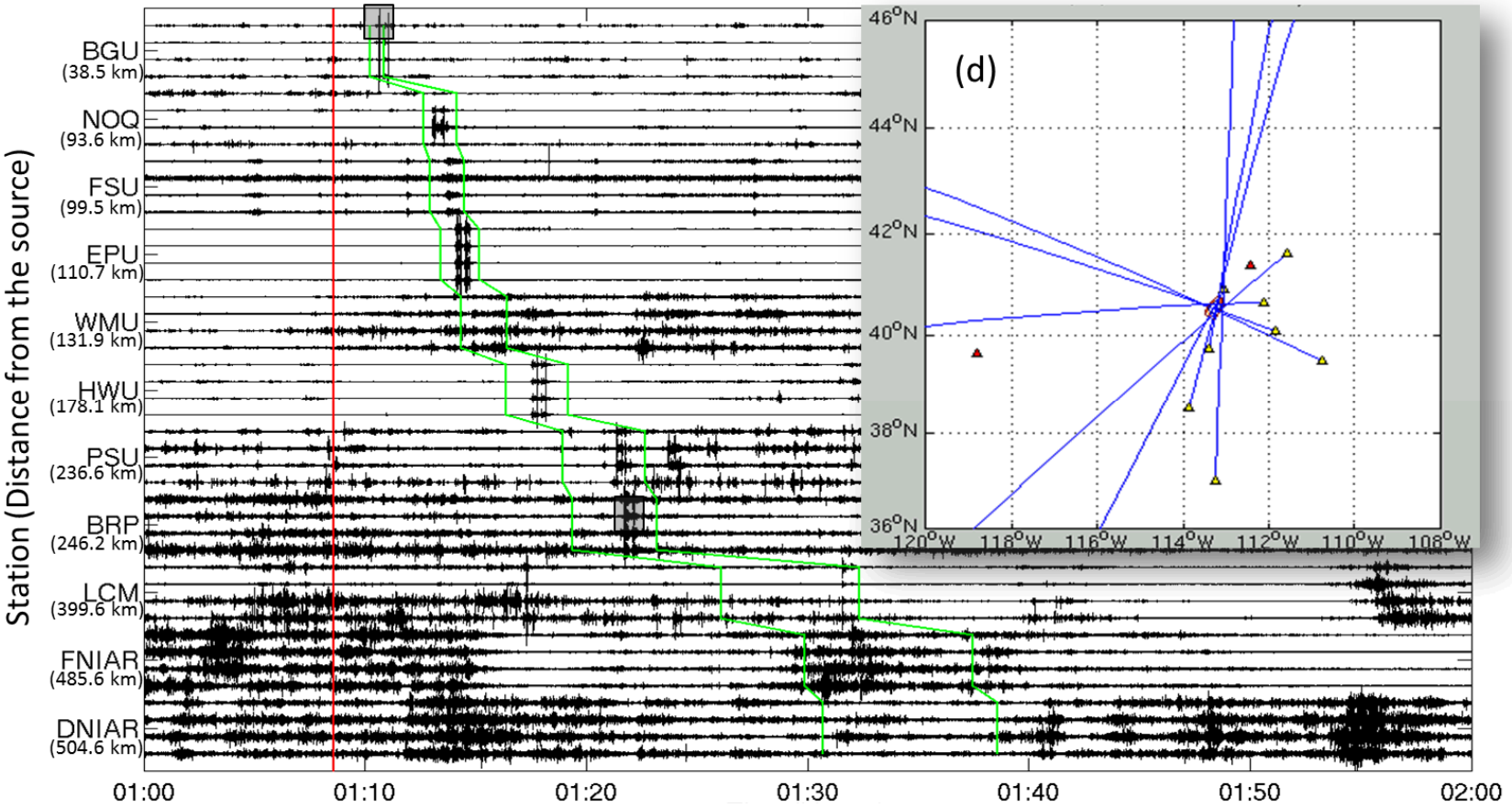
The importance of propagation modeling



Generating automatic infrasound catalogs

Regional detection and association with Bloodhound

Location of Utah source



Full-wave yield estimation

Bloodhound's yield estimation module uses state-of-the-art methods that produce the most robust estimates for above-ground explosions

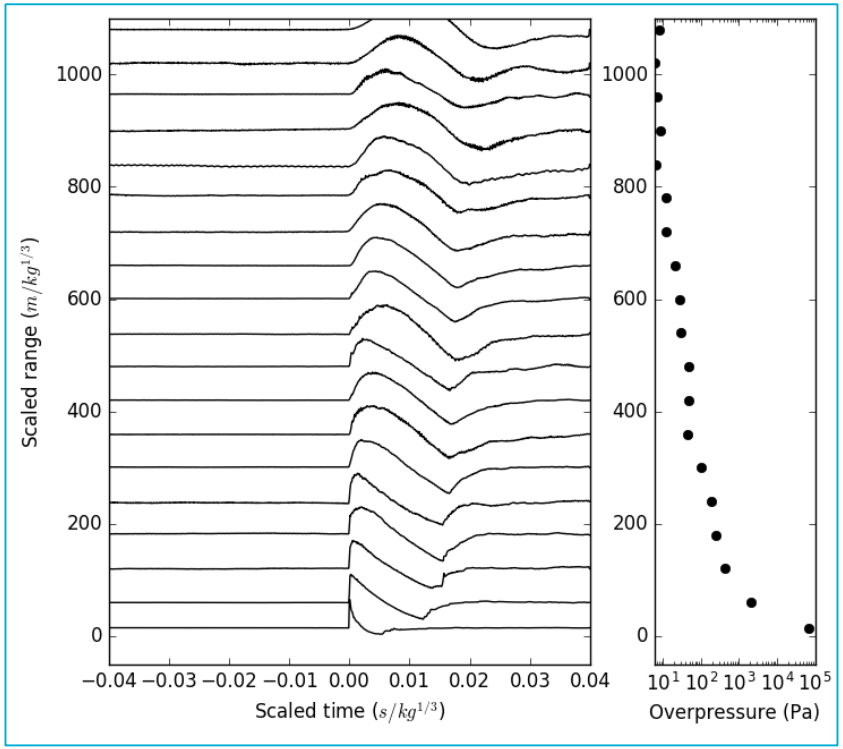
Why BLOODHOUND?

Existing methods measure overpressure and then relate to yield through empirical equations.

Bloodhound uses the whole waveform, resulting in much more accurate yields.

Status

The yield estimation module is not currently integrated into BLOODHOUND's pipeline processing system. Future research will merge this module into the pipeline.



The evolution of the blast wave as a function of scaled time and range is captured from historical observations and used to estimate the yield of unknown explosions.

Infrasound propagation simulations

Bloodhound contains built-in propagation codes and interfaces to community codes

Why BLOODHOUND?

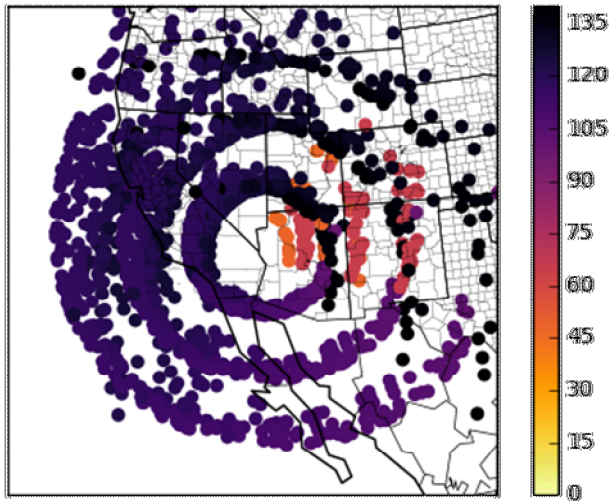
The current standard features ad hoc propagation tools with different interfaces.

Bloodhound has a single interface for multiple propagation tools, including built-in Python ray tracing that exploits multithreading.

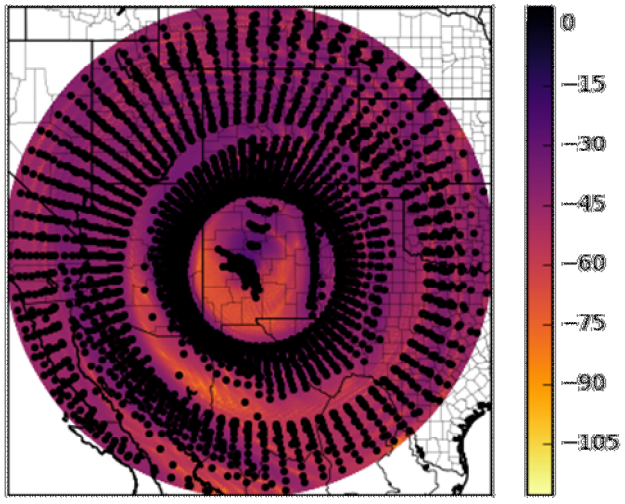
Status

Future research will focus on fully exploiting propagation tools in order to improve pipeline processing.

EXAMPLE SIMULATIONS



Ray bounce points from a source color-coded by maximum altitude



Transmission loss and bounce points for a different source

DEMO

Using Bloodhound to find events in Korea

Bloodhound is currently command-line based and relies on an input parameters specified by an SME.

Ultimately, we would like to more comprehensively integrate Bloodhound with Inpulse & Ephors to suggest parameters (that are easily tunable) based on the arrays selected

```
'''
Example input parameter file for bloodhound
'''

# --- Network parameters ---

arrays = ['BRDAR','CHNIAR','KSGAR','I45RU','I30JP']
array_lats = [37.965627, 38.271125, 38.589855, 44.1999016, 35.3077545]
array_lons = [124.64432, 127.12096, 128.35618, 131.977295, 140.313766]

# --- Array processing parameters ---

# slowness grid: X min, X max, Y min, Y max, Slow Step (Units in s/km):
sll_x=-3.6; slm_x=3.6; sll_y=-3.6; slm_y=3.6; sl_s=0.18

# Frequency properties:
frqlow=[0.01,0.5]; frqhigh=[0.5,3.]; win_len=[60.,40.]; win_frac=[.5,.5]

# --- Detector parameters ---

adaptive_timewindow = 3600.; T_v = 180.
p_thres = 0.01

# --- Association parameters ---

azimuth_dev = 10.
v_min = 0.22; v_max = 0.34

# --- Locator parameters ---

confidence_value = 0.9
sigma2 = (3.5 * np.pi / 180.) ** 2

# --- Additional parameters ---

Nproc = 10  # Number of threads to use
```

DEMO

Using Bloodhound to find events in Korea

```
import bloodhound as bh; import infra_params
bh.infra.process_data(infra_params, '2016.db', True, True, True, grid_file='korea_grid.p', fk_window=3600.)
```

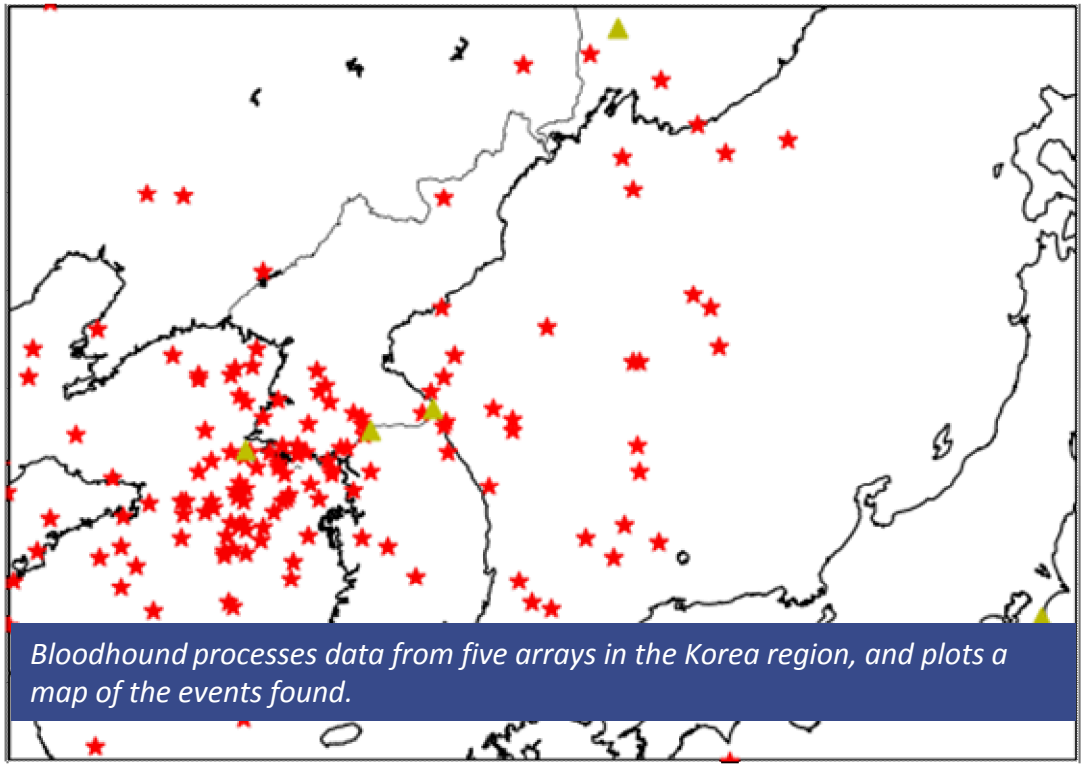
Output SQLite database file

Flags to indicate what steps to do

Input node locations

```
import infra_params
plot_map(3, '2016.db', infra_params, [33., 44.5, 120., 141.])
```

Minimum number of arrays in an event



Bloodhound processes data from five arrays in the Korea region, and plots a map of the events found.

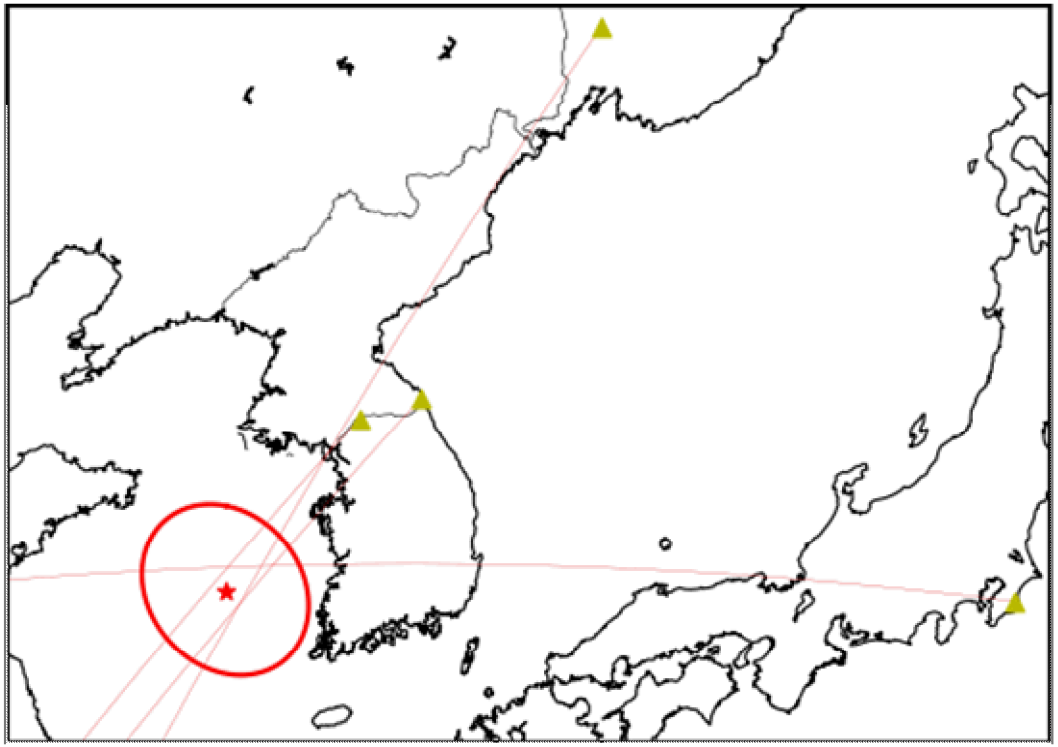
DEMO

Using bloodhound to find events in Korea

```
import bloodhound as bh
fig, ax = plt.subplots()
bh.infra.plot_origin_from_db('2016.db', 1, arrays=infra_params.arrays,
                             array_lats=infra_params.array_lats,
                             array_lons=infra_params.array_lons,
                             bounds=[33., 44.5, 120., 141.])
```

ORID

Geographic bounds for plot



Viewing a particular event association and location (identified by an ORID).

DEMO

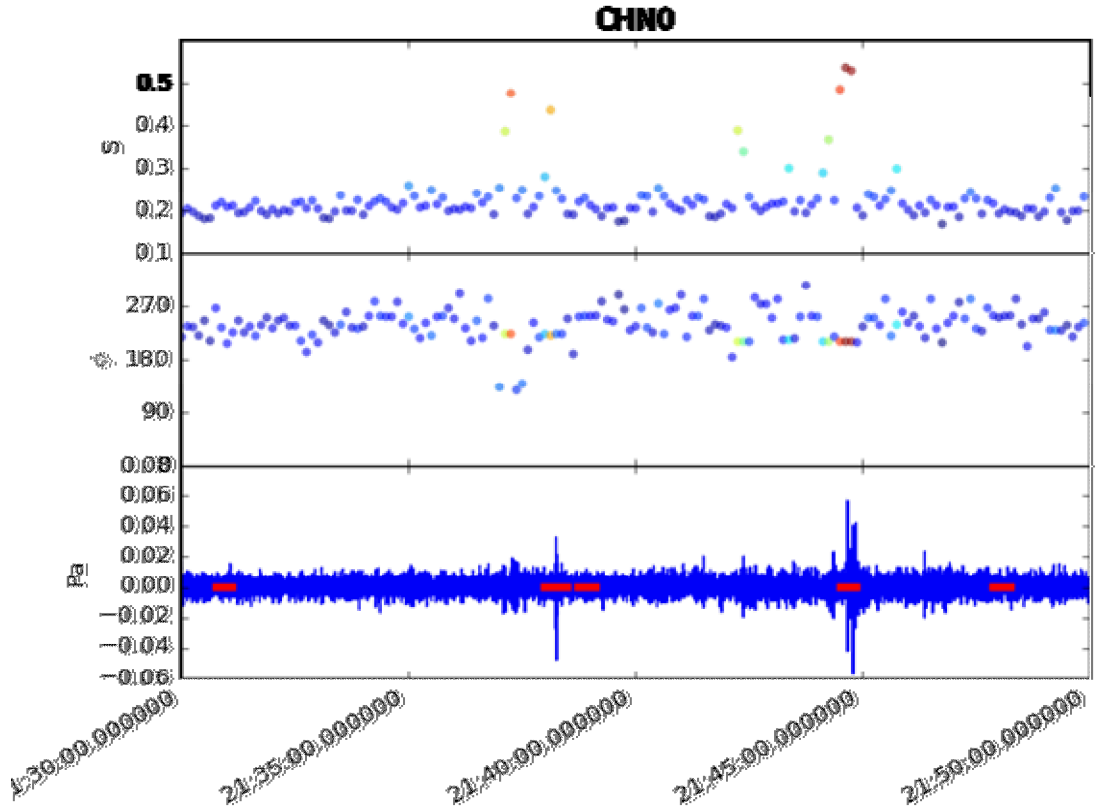
Using bloodhound to find events in Korea

```
%pylab notebook
import bloodhound as bh; from obspy import UTCDateTime; import datetime
t1 = UTCDateTime(datetime.datetime(2016,4,5,21,30,0))
t2 = UTCDateTime(datetime.datetime(2016,4,5,21,50,0))
bh.infra.plot_detect_from_db('2016.db', 'CHNAR', 0, [1., 5.], t1, t2)
```

Flag for detections to plot

Filter band

Start/end times for plot



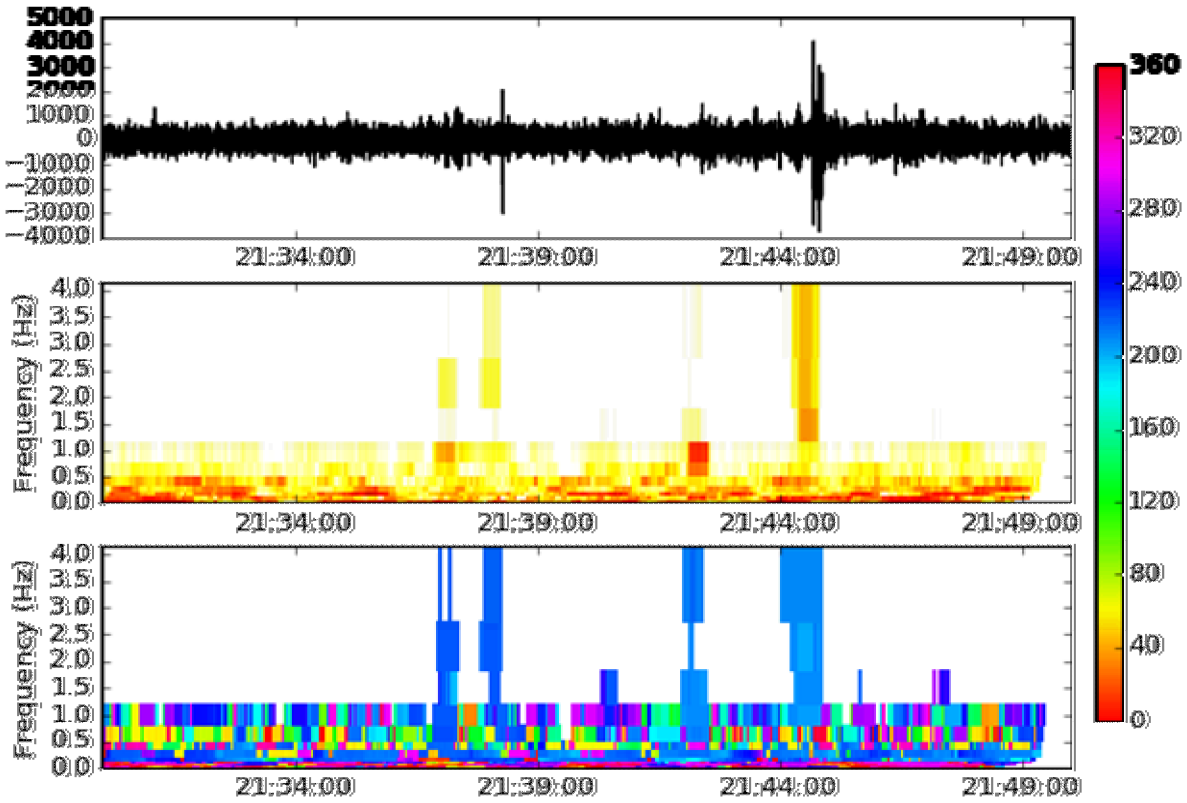
Plotting the detections associated with Event 1

DEMO

Using bloodhound to find events in Korea

```
import bloodhound as bh; from obspy import UTCDateTime; import datetime
t1 = UTCDateTime(datetime.datetime(2016,4,5,21,30,0))
t2 = UTCDateTime(datetime.datetime(2016,4,5,21,50,0))
st = bh.data.getStreamFromOracle(t1, t2, 'CHNAR')
st = st.select(channel='BDF')
bh.tools.pmcc(st, s_thres=0.3, plot_baz=True, log_scale=False, f_min=1., f_max=5.)
```

Semblance threshold

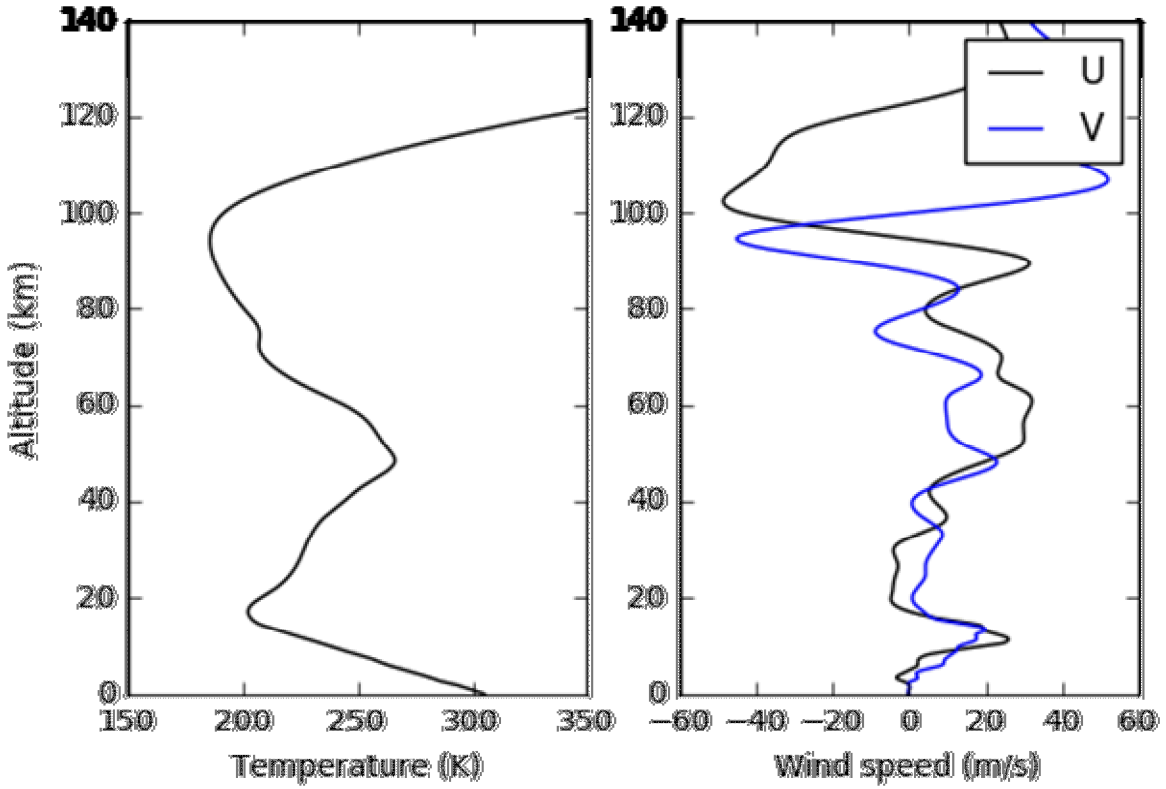


Exploring the time-frequency properties of the coherent energy.

DEMO

Propagation modeling with bloodhound

```
import bloodhound as bh
bh.propagate.plot_met('2016092818.met', xlim_temp=[150.,350.],
                      xlim_wind=[-60.,60.], ylim=[0.,140.])
```



Plotting a meteorological profile for propagation modeling.

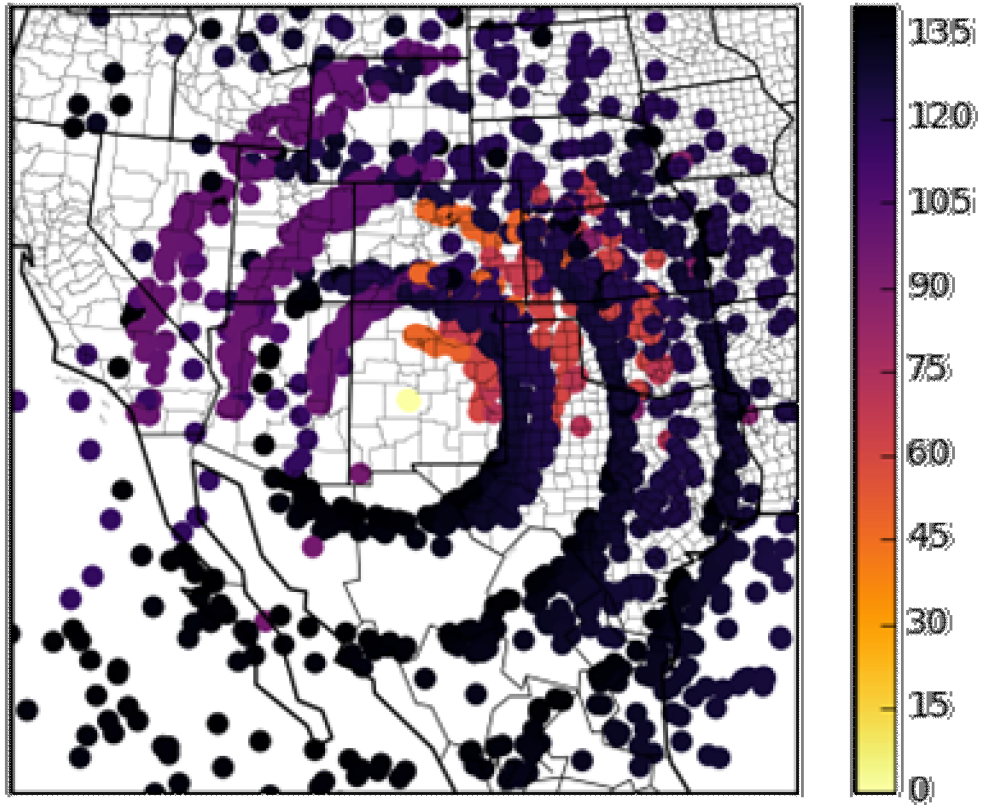
DEMO

Propagation modeling with bloodhound

```
import bloodhound as bh
rays = bh.propagate.taup_nby2d(-106.973665, 34.091126, '2016092818.met', 1000)
```

Event location

No. of rays to shoot



Running raytracer and plotting ray bounce points, color-coded by the maximum altitude of the ray

DEMO

Propagation modeling with bloodhound

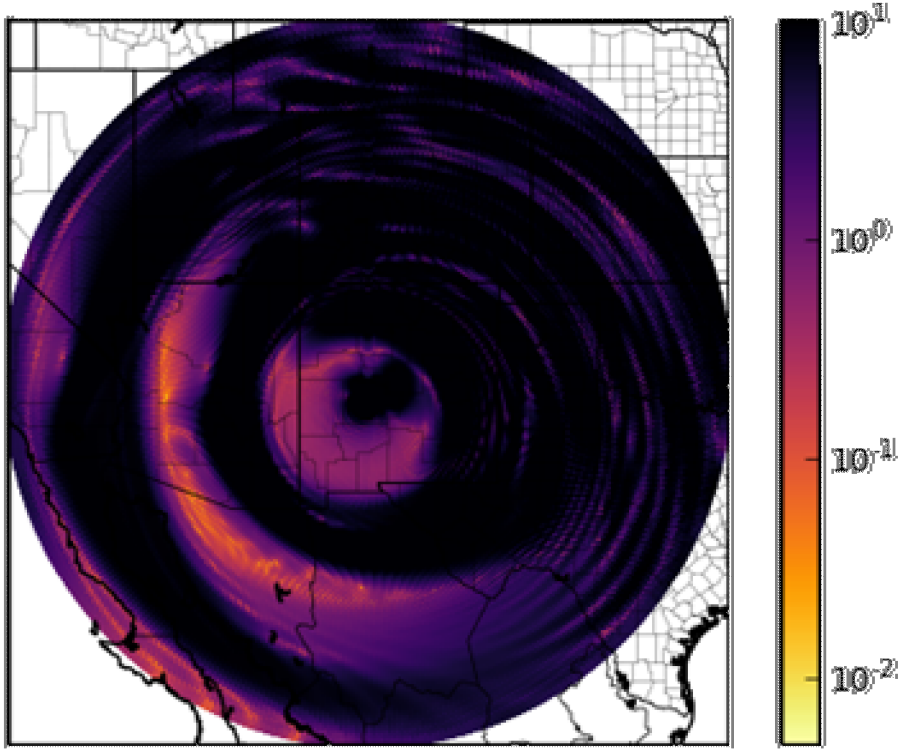
```
import bloodhound as bh

evla = 34.091126; evlo = -106.973665; evel = 1.4
W = 2400.       # Yield in kg
P0 = 866.       # Atmospheric pressure at shot
T0 = 23.        # Atmospheric temperature at shot
met_file = '2016092818.met'

bh.propagate.modess_nby2d_yield(evla, evlo, 0.1, met_file, W, P0, T0, evel=1.4, cmax=10.)
```

Frequency

Source parameters



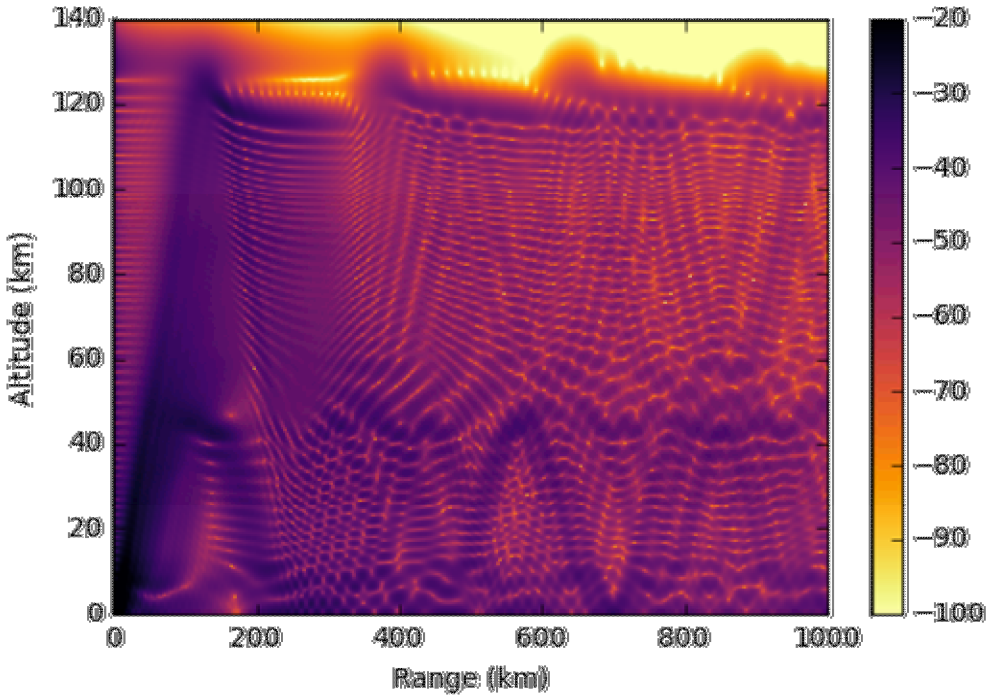
Running full wave code and plotting predicted amplitude as a function of location for a source of yield=2400 kg.

DEMO

Propagation modeling with bloodhound

```
import bloodhound as bh

az_source_to_usie = 75.48
range_source_to_usie = 331.2
usie_alt = 37.2
bh.propagate.modess_profile('2016092818.met', az_source_to_usie, 0.1, z_ground=1.4)
```



Running full wave code and plotting predicted amplitude as a function of altitude along a particular azimuth.