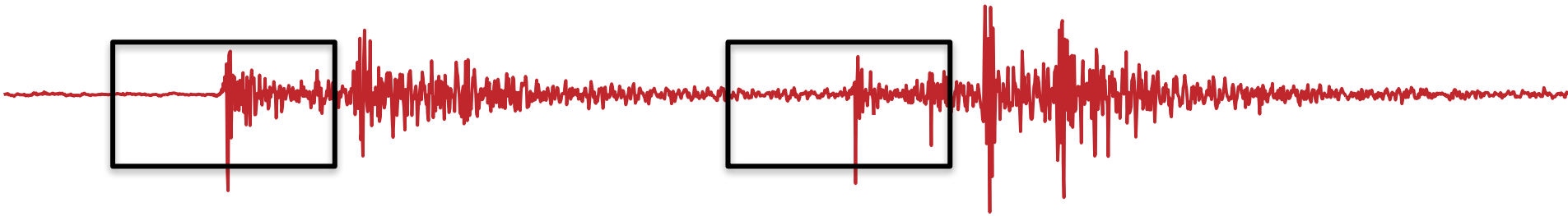
FAST TUTORIAL & USER GUIDE

Peter Bailis, Karianne Bergen, Gregory Beroza,
Hashem Elezabi, Philip Levis, Kexin Rong, Clara Yoon

Stanford University



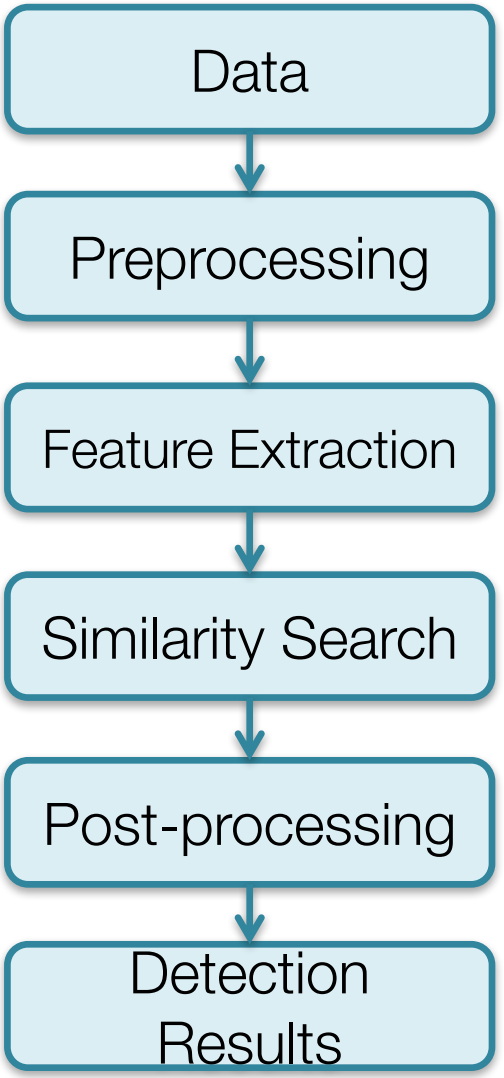
Code: https://github.com/stanford-futuredata/quake.git

Outline

1. FAST Overview
2. Requirements and Installation
3. Hector Mine Tutorial
4. How to set parameters
    1. Input and Preprocessing
    2. Fingerprint
    3. Similarity Search
    4. Network Detection
    5. Example Parameters
5. References
6. Supplementary

What is FAST?

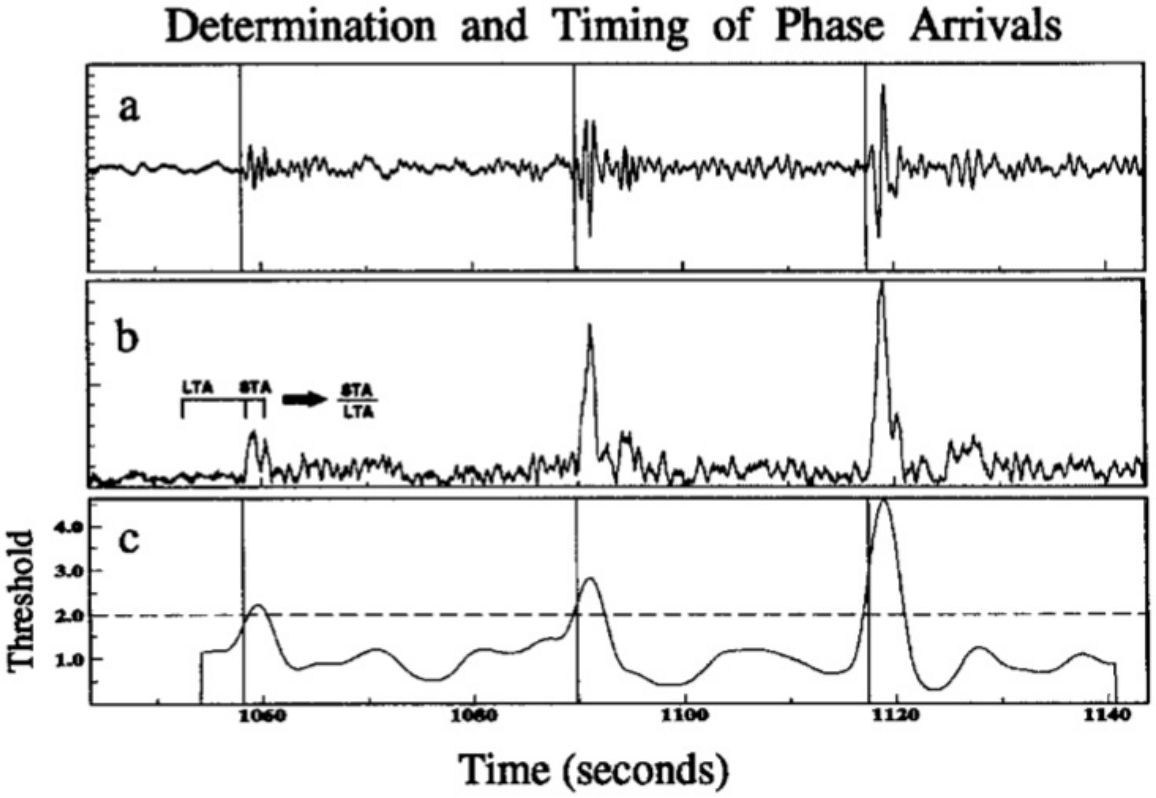
- End-to-end and unsupervised earthquake detection pipeline:
  - Input: continuous seismic station from multiple seismic stations
  - Output: A list of time stamps for potential detections
- Does not rely on catalogs of known events
- Designed to *complement* existing energy-based detection methods like STA/LTA



Standard Earthquake Detection: STA/LTA

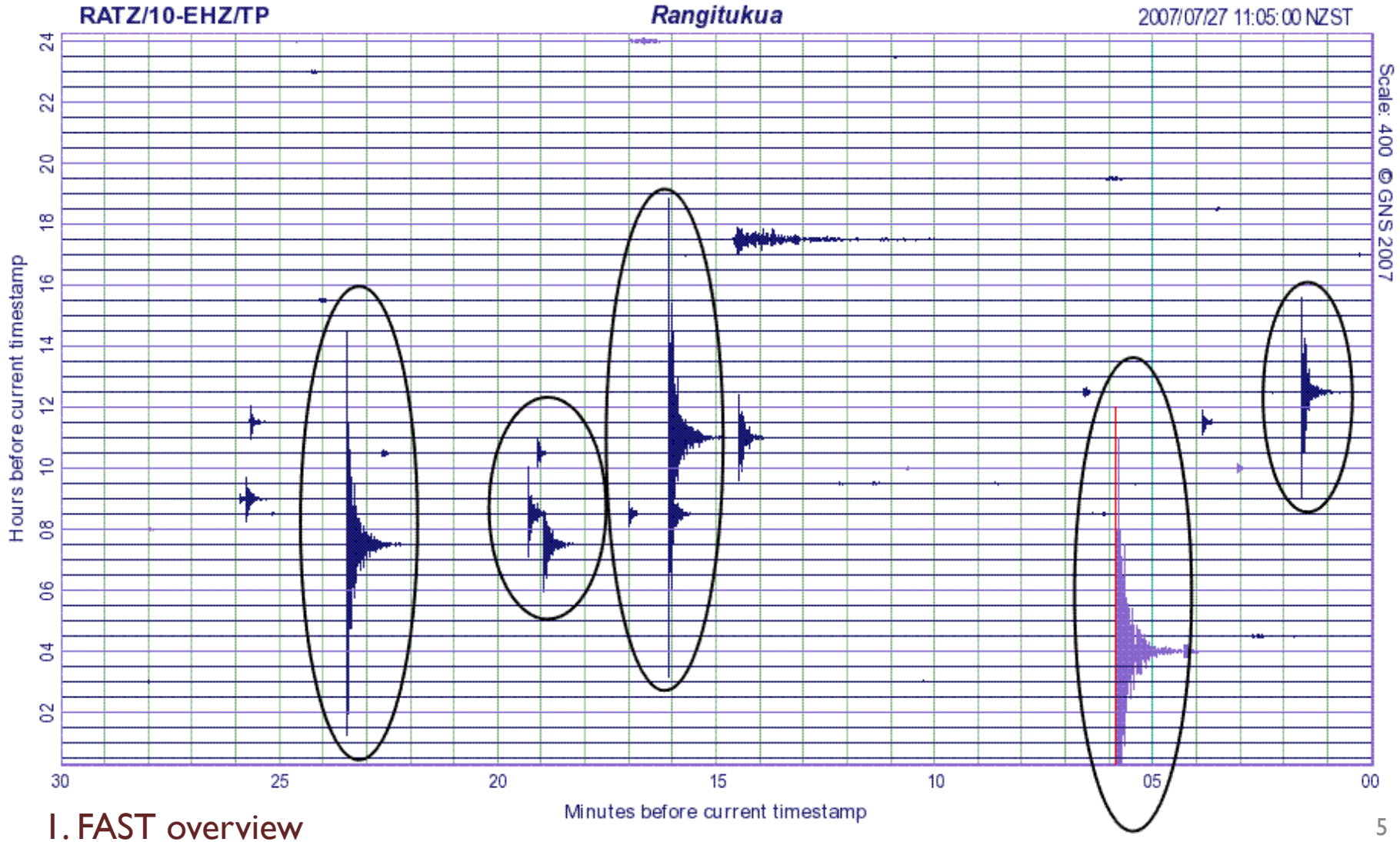
Ratio: Short Term Average / Long Term Average

- Energy based detector: impulsive P & S arrivals
- 1 station at a time
- Association at multiple stations



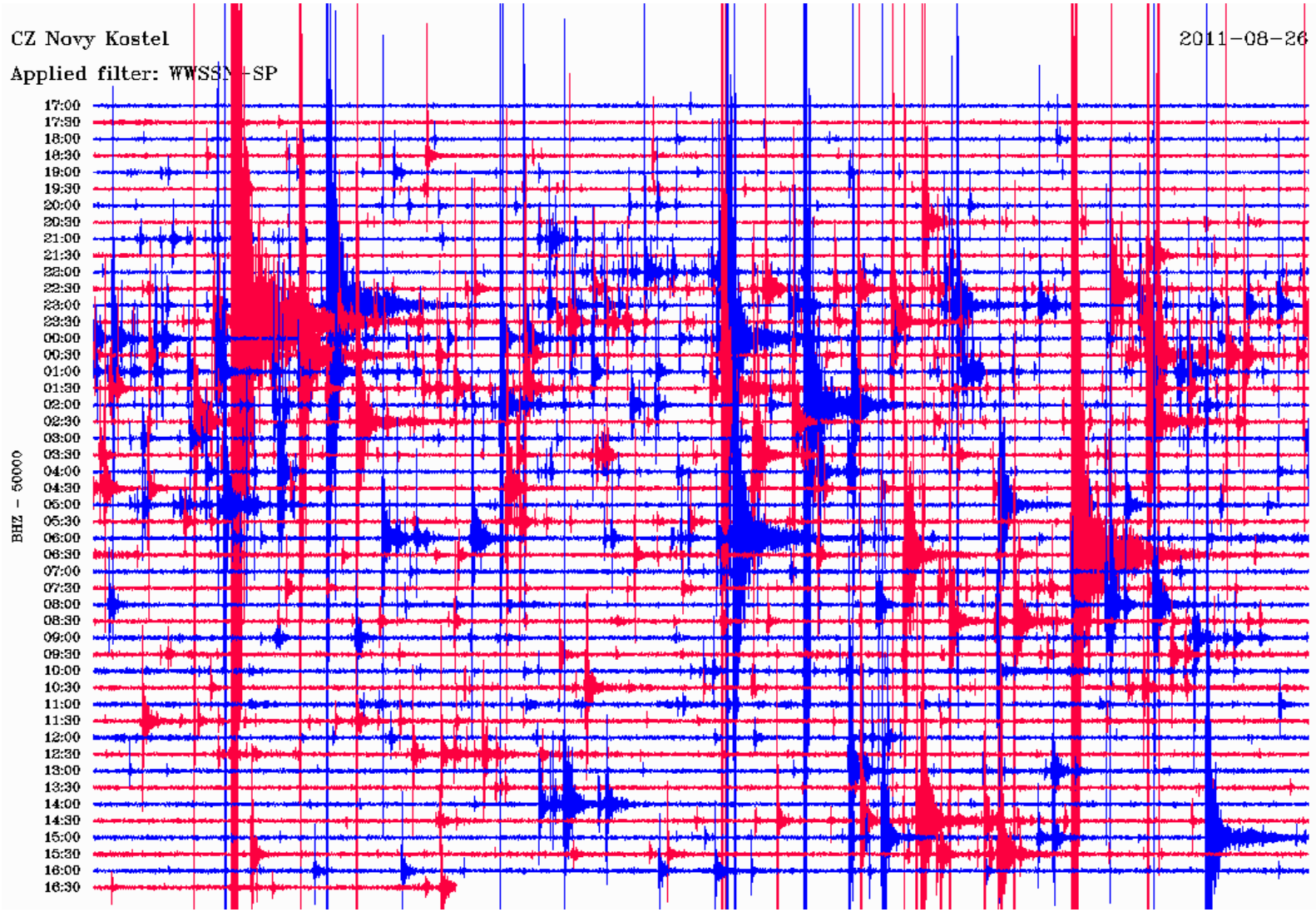
Earle and Shearer (1994)

STA/LTA earthquake detection works well for impulsive arrivals



http://info.geonet.org.nz/display/volc/Earthquakes

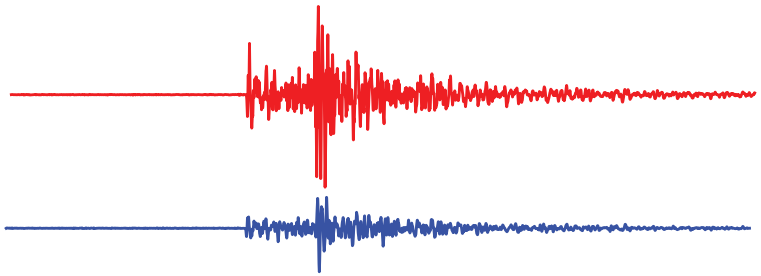
STA/LTA works less well for overlapping events, low snr scenarios etc.



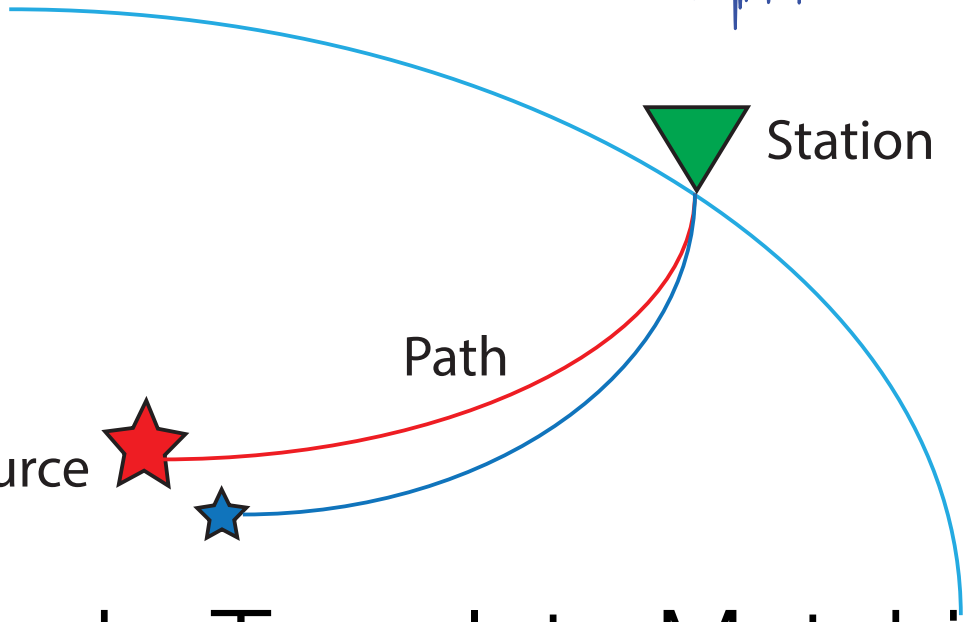
http://www.ig.cas.cz/userdata/pictures/observatories/webnet/aktualni/NKC.active5.gif

Waveform similarity allows detection of smaller earthquakes

Supervised



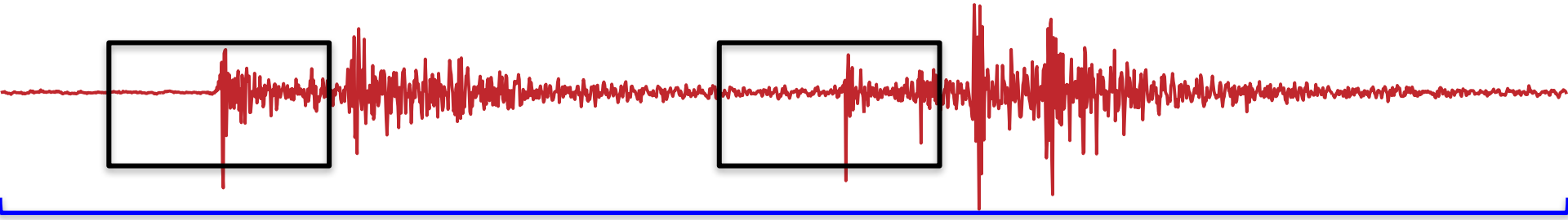
Template waveform:
known
Similar event waveform:
previously unknown



Informed search: Template Matching

Comprehensive, exhaustive search for earthquakes with similar waveforms

Unsupervised



10 years

Treat each segment of the input waveform data as a "template" for potential earthquakes.

Uninformed (blind) search

Efficient search for similar items in large databases

Music

Webpages

Videos

Fingerprint And Similarity Thresholding (FAST)

Earthquakes

What can and can't FAST do?

- Typical use case: weeks/months/years of continuous seismic data at several stations, scan through it to find small similar (or repeating) earthquakes

- Limitation: FAST will not detect an earthquake that occurs **only once** and is **not similar enough** to any other earthquakes in your continuous data set

- Most similar earthquakes != largest earthquakes
  - Larger earthquakes often match a smaller earthquake on the coda, so they are detected with lower similarity (or not detected at all)

- FAST is intended to complement (not replace) existing energy-based detection methods like STA/LTA

Outline

1. FAST Overview
2. Requirements and Installation
3. Hector Mine Tutorial
4. How to set parameters
    1. Input and Preprocessing
    2. Fingerprint
    3. Similarity Search
    4. Network Detection
    5. Example Parameters
5. References
6. Supplementary

Requirements

- FAST is written in Python and C++ and is designed to run on Linux clusters.
- Install instructions might not work for Windows and macOS machines.
- The code will benefit from running on machines with more memory and CPUs.
  - Consider using instances from Amazon AWS or Google Cloud

Install

- Download the FAST code from Github

```
git clone https://github.com/stanford-futuredata/quake.git
```

- Install Python dependencies

```
pip install -r requirements.txt
```

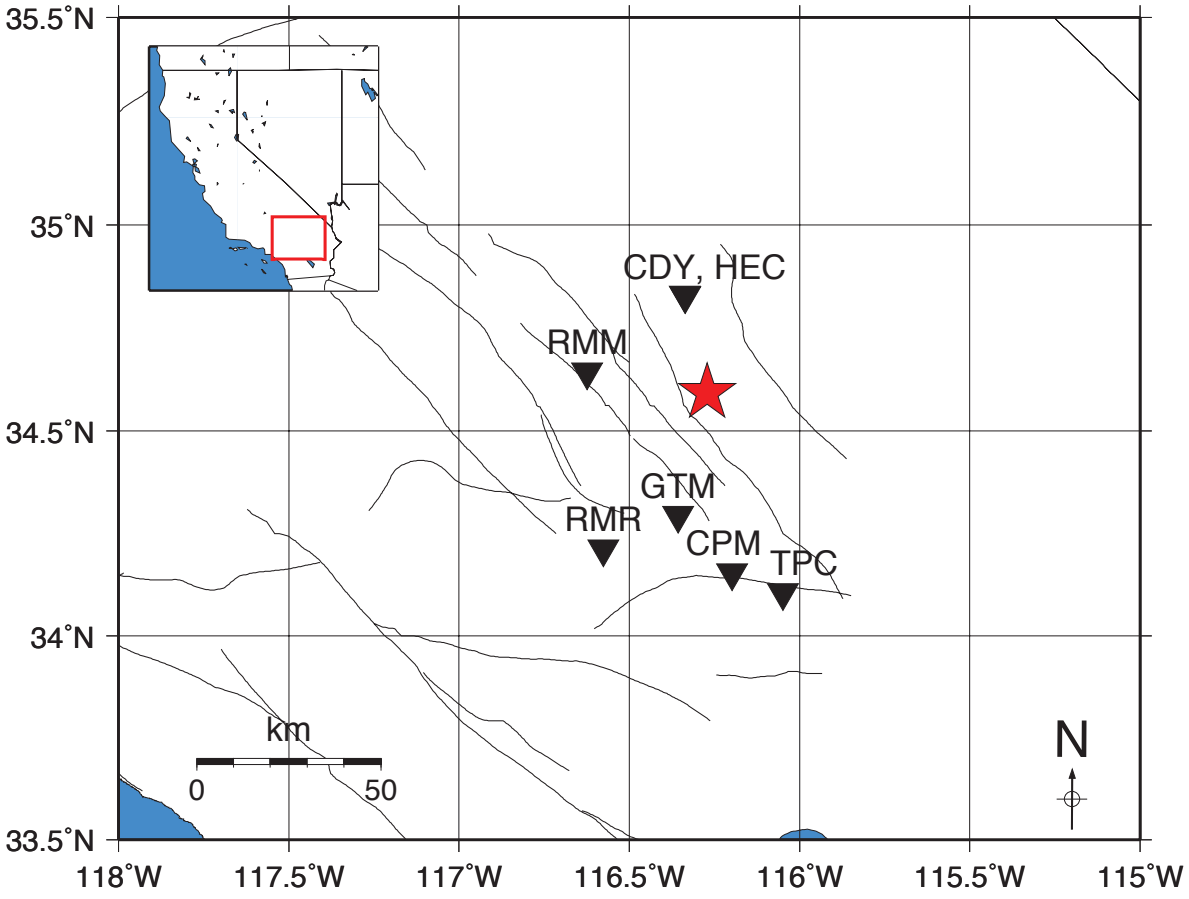
- Install C++ dependencies

```
sudo apt-get install cmake
sudo apt-get install build-essential
sudo apt-get install libboost-all-dev
```

Outline

1. FAST Overview
2. Requirements and Installation
3. Hector Mine Tutorial
4. How to set parameters
    1. Input and Preprocessing
    2. Fingerprint
    3. Similarity Search
    4. Network Detection
    5. Example Parameters
5. References
6. Supplementary

Example Data: Hector Mine Foreshocks



20 hours continuous data from 7 stations (9 components) Already bandpass filtered, decimated to 20 Hz

HEC is 3-component; Other stations 1-component

waveform data: `data/waveforms${STATION}/Deci5*`

File structure overview

Code

- Fingerprint: `fingerprint/`
- Similarity Search: `simsearch/`
- Postprocessing: `postprocessing/`
- Utility Functions :
  - `utils/preprocess`
  - `utils/network`
  - `utils/events`
  - `run_fp.py`
  - `run_simsearch.py`

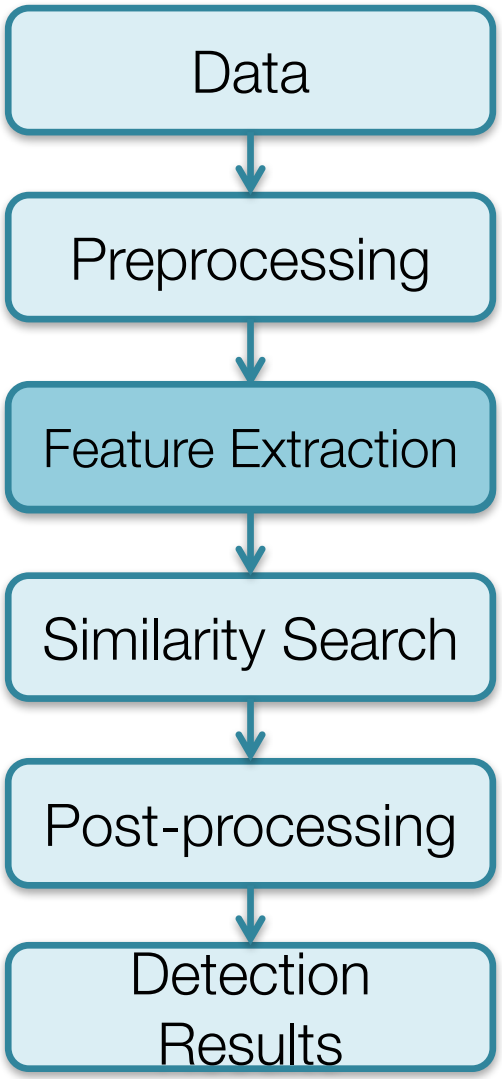
Configuration and parameters

- `parameters/`
  - `fingerprint/`
  - `simsearch/`
  - `postprocess/`

Data

- `data/waveforms${STATION}/Deci5*`

Feature Extraction



Input

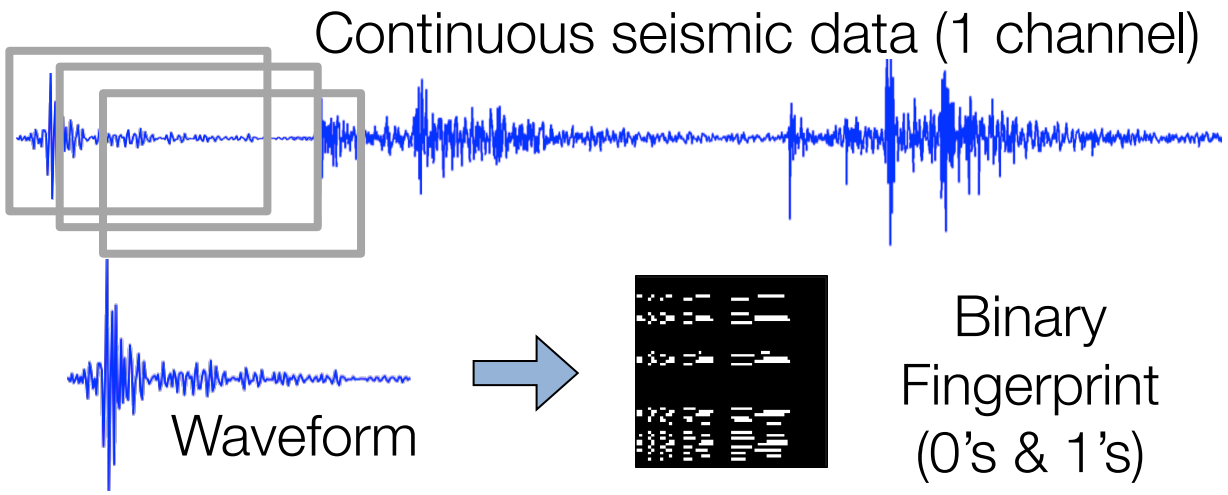
continuous seismic data from a channel

Output

a binary fingerprint for each overlapping segment of the original waveform

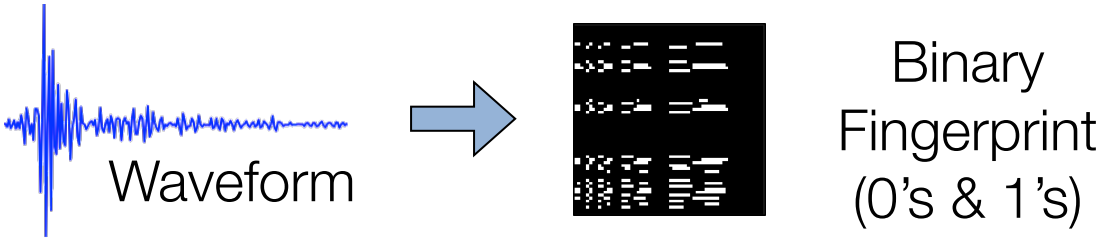
Key Property

Similarity of the binary fingerprints approximates that of the original waveforms



Generate fingerprints

- Create fingerprints for each of 9 channels (7 stations) + global index, using wrapper
  - `~/quake_tutorial$ python run_fp.py –c config.json`
  - – All fingerprint input files in `parameters/fingerprint/`
- Alternatively, to fingerprint a specific station, call the fingerprint script with the corresponding fingerprint parameter file:
  - `~/quake_tutorial$ cd fingerprint/`
  - `~/quake_tutorial/fingerprint$ python gen_fp.py ../parameters/fingerprint/fp_input_CI_CDY_EHZ.json`

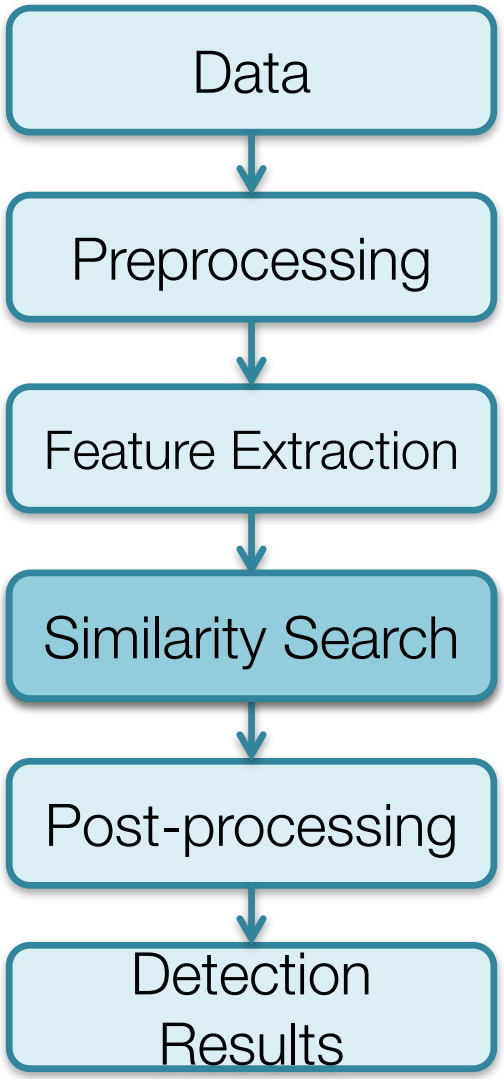


Global Index

*** NOTE: already called by run_fp.py wrapper

- Complete this step only after you have finished computing fingerprints for every component and station you want to use for detection
  - $ `python global_index.py global_indices.json`
  - `global_index.py` in `fingerprint/`
  - `global_indices.json` in `parameters/fingerprint/`
- Continuous data start/end times may be different, and time gaps may happen at different times, at different components and stations
- Global index: consistent way to refer to times of fingerprints at different components and stations

Similarity Search



Input

binary fingerprints from one seismic channel

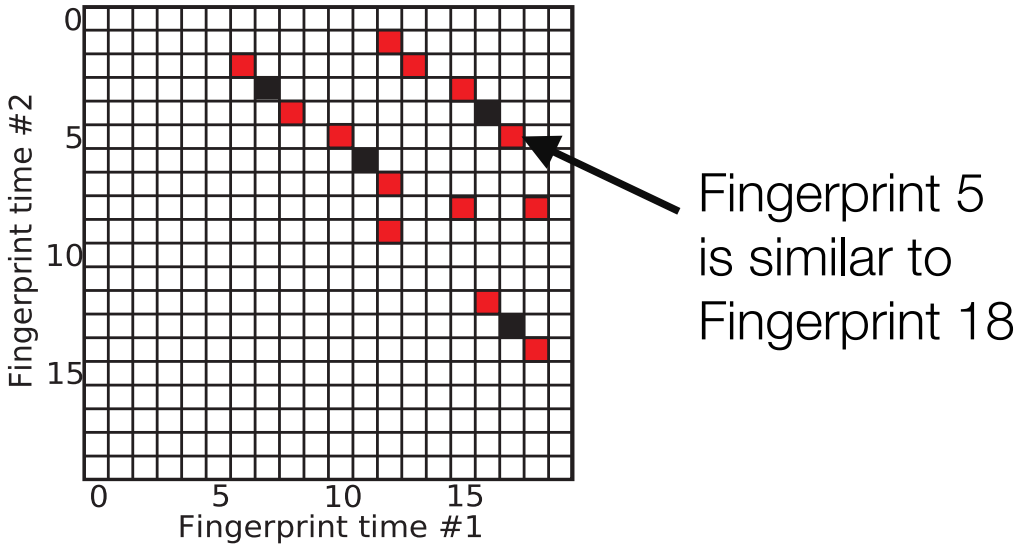
Output

all pairs of binary fingerprints whose (Jaccard) similarity is above the threshold

Efficient approximate similarity search

MinHash (probabilistic estimate of Jaccard similarity)

Locality Sensitive Hashing (LSH)



Search for similar earthquake pairs

- Compile and build C++ similarity search code
  - `~/quake_tutorial$ cd simsearch/`
  - `~/quake_tutorial/simsearch$ cmake .`
  - `~/quake_tutorial/simsearch$ make`

- Similarity search for each of 9 channels (7 stations), using wrapper
  - `~/quake_tutorial/simsearch$ cd ..`
  - `~/quake_tutorial$ python run_simsearch.py –c config.json`

- Alternatively, to fingerprint a specific station, call the fingerprint script with the corresponding fingerprint parameter file:
  - `~/quake_tutorial$ cd simsearch/`
  - `~/quake_tutorial/simsearch$ cp ../parameters/simsearch/simsearch_input_HectorMine.sh .`
  - `~/quake_tutorial/simsearch$ ./simsearch_input_HectorMine.sh`

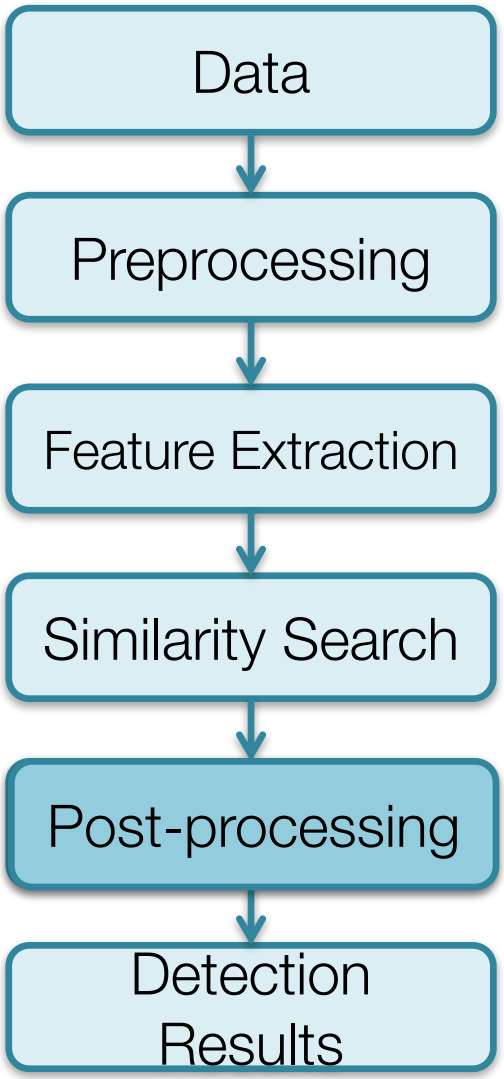
FAST Similarity Search Output (1 Channel)

- `data/waveforms${STATION}/fingerprints/`
  - Min-Hash Signatures (can delete these later)
    - `mh_${STATION}_${CHANNEL}_${nhash}.bin`
    - Example: `mh_CDY_EHZ_4.bin`
  - Binary files with similarity search output (npart files, one per partition, with first and last fingerprint index for the partition in filename):
    - `candidate_pairs_${STATION}_${CHANNEL}_${nhash},${ntbl}(${FIRST_FP_INDEX},${LAST_FP_INDEX})`
    - Example: `candidate_pairs_CDY_EHZ_4,2(0,74793)`
    - For efficiency, the output is binary format; need parsing to convert similarity search output to text files

Parse FAST Similarity Search Output: [Binary → Text File]

- Use the wrapper script to parse all 9 channels (7 stations)
  - `~/quake_tutorial$ cd postprocessing`
  - `~/quake_tutorial/postprocessing$ cp ../parameters/postprocess/*.sh .`
  - `~/quake_tutorial/postprocessing$ ./output_HectorMine_pairs.sh`
- Parse a specific channel
  - Syntax: `python parse_results.py –d <folder_with_binary_sim_search_files> –p <sim_search_filename_prefix> –i <global_index_file>`
  - Example input file: `candidate_pairs_CDY_EHZ_4,2(0,74793)`
  - Example command: `$ python parse_results.py –d ../data/waveformsCDY/fingerprints/ candidate_pairs_CDY_EHZ –i ../data/global_indices/CDY_EHZ_idx_mapping.txt`

Postprocessing

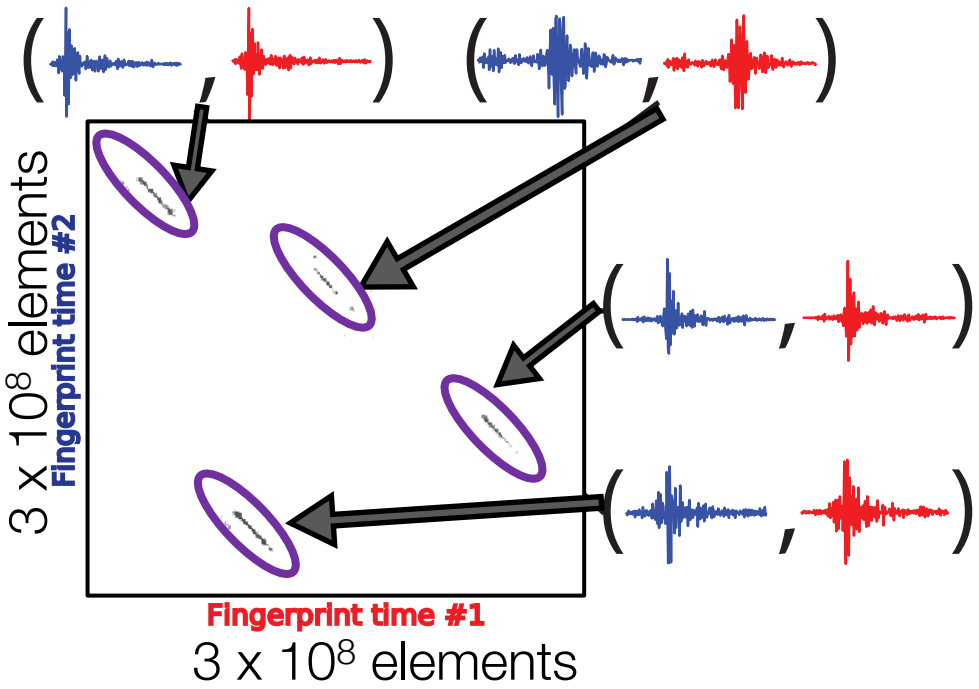


Input

Similarity search output in the form of a sparse similarity matrix

Output

Groups in the matrix corresponding to potential earthquake detections



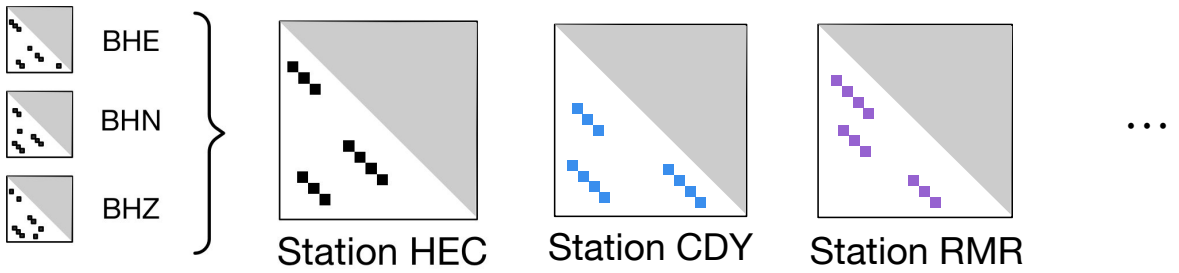
Postprocess: Combine FAST Similarity Search Output (3 components @ 1 station)

- Combine similarity matrix from all components in each station, and copy outputs to `../data/inputs_network/` using wrapper script:
  - `~/quake_tutorial/postprocessing$ cp ../parameters/postprocess/*.sh .`
  - `~/quake_tutorial/postprocessing$ ./combine_HectorMine_pairs.sh`

- Alternatively, to combine outputs (add "FAST similarity" for same fingerprint pair (index1, index2)) from a specific station:
  - `$ python parse_results.py –d <folder_with_text_sim_search_files> –p <text_sim_search_file_prefix> --sort true –parse false –c true –t <threshold>`
  - Example: $ `python parse_results.py –d ../data/inputs_network/ –p candidate_pairs_HEC --sort true –parse false –c true –t 6`
    - Adds FAST similarity for same fingerprint pairs on HEC components: `candidate_pairs_HEC_BHE_merged.txt, candidate_pairs_HEC_BHN_merged.txt, candidate_pairs_HEC_BHZ_merged.txt`
    - Output (one file per station): `candidate_pairs_HEC_combined.txt`
      - Warning: By default, the *merged.txt files for all 3 components are deleted after combining! Copy them before this step if you want to keep them.
  - Usually set **threshold**=number of components * v, where v=nvote at a single component
    - The threshold helps filter out matches generated from noise, since we require either strong matches at a single component, or weak matches at multiple components
    - May want to set threshold slightly lower (e.g., 2*v) if the output size is too small after combining

Postprocess: weight stations equally

- 'Equalize' across network: weighting stations with different number of components equally
  - Hector Mine example: one 3-component station (HEC) with similarity threshold at 6 votes, and 6 1-component stations with similarity threshold at 2 votes.
  - Want to weight each station equally, so multiply similarity in each 1-component station by 3
    - `$ awk '{print $1, $2, 3*$3}' candidate_pairs_CDY_EHZ_merged.txt > candidate_pairs_CDY_combined.txt`
- Another option: use only part of the data (e.g. vertical component at each station)



Postprocess: Network Detection

- Run network detection (combine FAST results from all 7 stations):
  - `~/quake_tutorial/postprocessing$ cp ../parameters/network/* .`
  - `~/quake_tutorial/postprocessing$ python scr_run_network_det.py 7sta_2stathresh_network_params.json`

Input file: `7sta_2stathresh_network_params.json`

```
"io": {
        "channel_vars": ["CDY", "CPM", "GTM", "HEC", "RMM", "RMR", "TPC"],   Station names
        "fname_template": "candidate_pairs_%s_combined.txt",   Input file (to fill w/station name)
        "base_dir": "../data/",   Base directory
        "data_folder": "inputs_network/",   Input file directory
        "out_folder": "network_detection/"   Output file directory
    }
```

Network Detection Outputs

- Network Detection Output (text file with labeled columns)
  - Example (ranked in descending order of 'peaksum'): `7sta_2stathresh_detlist_rank_by_peaksum.txt`
  - First (num_sta=number of stations) columns: starting fingerprint index at each station (time information)
    - Outputs "nan" if not observed at a particular station
  - dL: Maximum length (samples) along diagonal, over all event-pairs containing this event
  - nevents: Number of other events 'linked' to (similar to) this event
  - nsta: Number of stations over which other events are similar to this event
  - tot_ndets: Total number of fingerprint-pairs (pixels) containing this event, over all event-pair clusters, over all stations
  - max_ndets: Maximum number of fingerprint-pairs (pixels) containing this event, over all event-pair clusters, over all stations
  - tot_vol: Total sum (or 'volume') of all similarity values (added over all stations), over all event-pairs containing this event
  - max_vol: Maximum sum (or 'volume') of all similarity values (added over all stations), over all event-pairs containing this event
  - max_peaksum: Maximum similarity value (added over all stations), over all event-pairs containing this event

Example custom scripts to clean up and visualize network detection results

- Depending on the parameters, network detection output text file might still have duplicate events
- We provide some example post-processing scripts to remove these duplicate events and come up with a final list of event detections
  - `cp ../utils/network/* .`
  - Note: these have not been fully tested. You may want to write your own instead!
  - Need to modify input/output parameters within each script
  - 4 scripts: outputs of each script are inputs to the next script

Network Detection "Deduplication"

- 1) `~/quake_tutorial/postprocessing$ python arrange_network_detection_results.py`
  - Save only start and end fingerprint indices for each event (First num_sta columns → 2 columns)
  - Output 2 more columns at end
    - Num_sta: number of stations that detected event
    - Diff_ind: Difference between first and last fingerprint index
  - Example output: `NetworkDetectionTimes_7sta_2stathresh_detlist_rank_by_peaksum.txt`

- 2) `~/quake_tutorial/postprocessing$ ./remove_duplicates_after_network.sh`
  - Remove events with duplicate start fingerprint index (keep events with highest num_sta then peaksum)
  - Example output: `uniquestart_sorted_no_duplicates.txt`

- 3) `~/quake_tutorial/postprocessing$ python delete_overlap_network_detections.py`
  - Remove events with overlapping times: where start time of one event is before end time of another event
  - Example output: `7sta_2stathresh_FinalUniqueNetworkDetectionTimes.txt`

- 4) `~/quake_tutorial/postprocessing$ ./final_network_sort_nsta_peaksum.sh`
  - Sort events in descending order of num_sta (number of stations that detected event), then peaksum (maximum similarity for this event)
  - Should no longer have duplicate events
  - Example output: `sort_nsta_peaksum_7sta_2stathresh_FinalUniqueNetworkDetectionTimes.txt`

Final Thresholds: Visual/Manual Inspection

- Now we have a list of detections – but are they actually earthquakes? Need to plot and visually inspect them

- Example script, view first 100 events: `~/quake_tutorial/utils/events$ python PARTIALplot_hector_detected_waveforms.py 0 100`
  - Input fingerprint indices from `sort_nsta_peaksum_7sta_2stathresh_FinalUniqueNetworkDetectionTimes.txt`
  - Need global start time (t0) from global_idx_stats.txt, dt_fp in seconds
    - Here t0 = UTCDateTime('1999-10-15T13:00:00.676000')
  - Event time = t0 + dt_fp*(start fingerprint index)
  - Cut short time window around event waveform in filtered data, plot across all stations, save image to png file with names ordered in descending order of num_sta, peaksum – does it look like an earthquake?

- Set final thresholds for num_sta, peaksum
  - Example output (first 50 events): `EQ_sort_nsta_peaksum_7sta_2stathresh_FinalUniqueNetworkDetectionTimes.txt`

- Additional post-processing scripts to remove false positive detections?

Final Event Detection List

- Example Script: `~/quake_tutorial/utils/events$ python output_final_detection_list.py`
- Example Input: `EQ_sort_nsta_peaksum_7sta_2stathresh_FinalUniqueNetworkDetectionTimes.txt`
- Example Output: `FINAL_Detection_List_HectorMine_7sta_2stathresh.txt`
  - Column 1: YYYY-MM-DDTHH:MM:SS.SSSS for event detection time. This is an approximate arrival time for the event (NOT origin time)
  - Column 2: Event detection time (seconds) since the start time for the first fingerprint, which for this data set is UTCDateTime('1999-10-15T13:00:00.676000')
  - Column 3: First fingerprint index (integer) for this event. Multiply by dt_fp = 1 second to get the time in column 2. Columns 1, 2, 3 basically represent the same information.
  - Column 4: Last fingerprint index (integer) for this event from the network-detection
  - Column 5: Fingerprint index duration; column5 = column4-column3. Some sense of event duration
  - Column 6: Number of stations at which the event was similar enough for a detection, the higher the better (nsta_thresh = 2)
  - Column 7: Peak sum of similarity for this event over all stations where it was detected, the higher the better (more similar to some other event).
  - Detection list is ordered in descending order of number of stations (Column 6), then in descending order of peaksum similarity (Column 7).
- Further processing is required for P/S phase picking and location
  - Cut SAC files
  - Pick phases (automatic or manual)
  - Locate earthquakes
  - Compute magnitudes

Outline

1. FAST Overview
2. Requirements and Installation
3. Hector Mine Tutorial
4. How to set parameters
    1. Input and Preprocessing
    2. Fingerprint
    3. Similarity Search
    4. Network Detection
    5. Output
5. References
6. Supplementary

Input: Continuous seismic data

- SAC or MiniSEED formats
  - Run FAST independently for each component at each station
    - Filename should have station, component, date-time
  - Filename must end with ".sac" or ".mseed" – if not, rename it!
  - FAST uses ObsPy functions to read in data

- Time gaps ok, but must not be filled with 0's
  - 0's are similar to other 0's → time gaps overwhelm your earthquake detections, and take forever to run
  - Run scripts to fill 0's with random uncorrelated noise; outputs files starting with "Filled*"
  - $ `python fill_time_gaps_uncorrelated_noise.py` (calls `detect_time_gaps.py`) – need to modify for your input data
    - Example scripts in `parameters/preprocess_utils/`

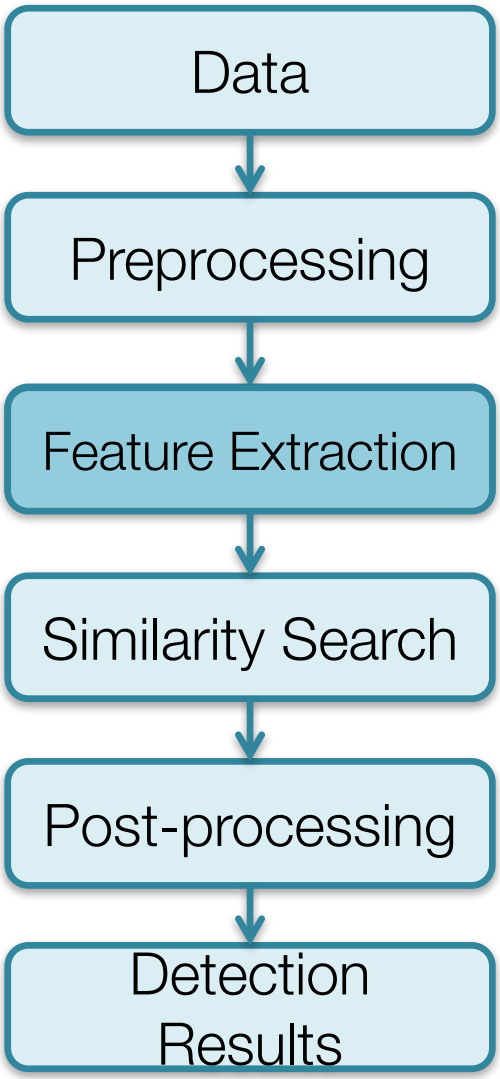
Preprocessing

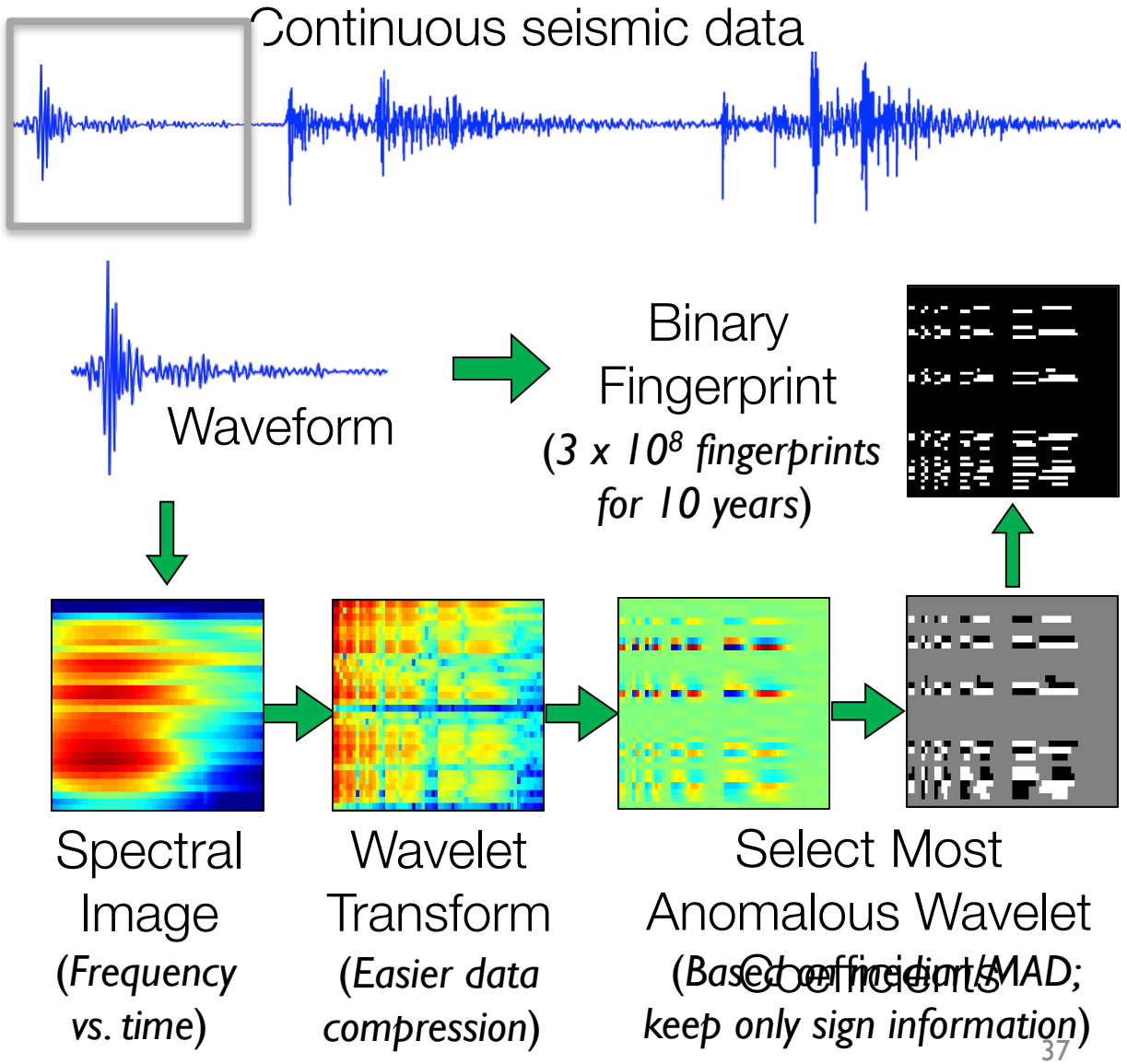
- Bandpass Filter: important parameter
  - FAST uses short sections of spectrogram to create fingerprints
    - Nyquist upper limit = (Sampling rate)/2
    - Typically sampling rate is 100 Hz, but depends on data
  - Frequencies outside passband are cut, thrown away
  - Demean, detrend data before applying filter
- Decimate to lower sampling rate
  - Depends on your selected filter band; usually to 50 Hz, 25 Hz, or 20 Hz
- Remove spikes and glitches (write your own script)
- Example Python scripts in `parameters/preprocess_utils/` (uses ObsPy):
  - `bandpass_filter_decimate.py, mseed_bandpass_filter_decimate.py`
  - `SaudiMonth_bandpass_filter_decimate.sh`: calls Python script multiple times for different stations and components, different filter bands

Outline

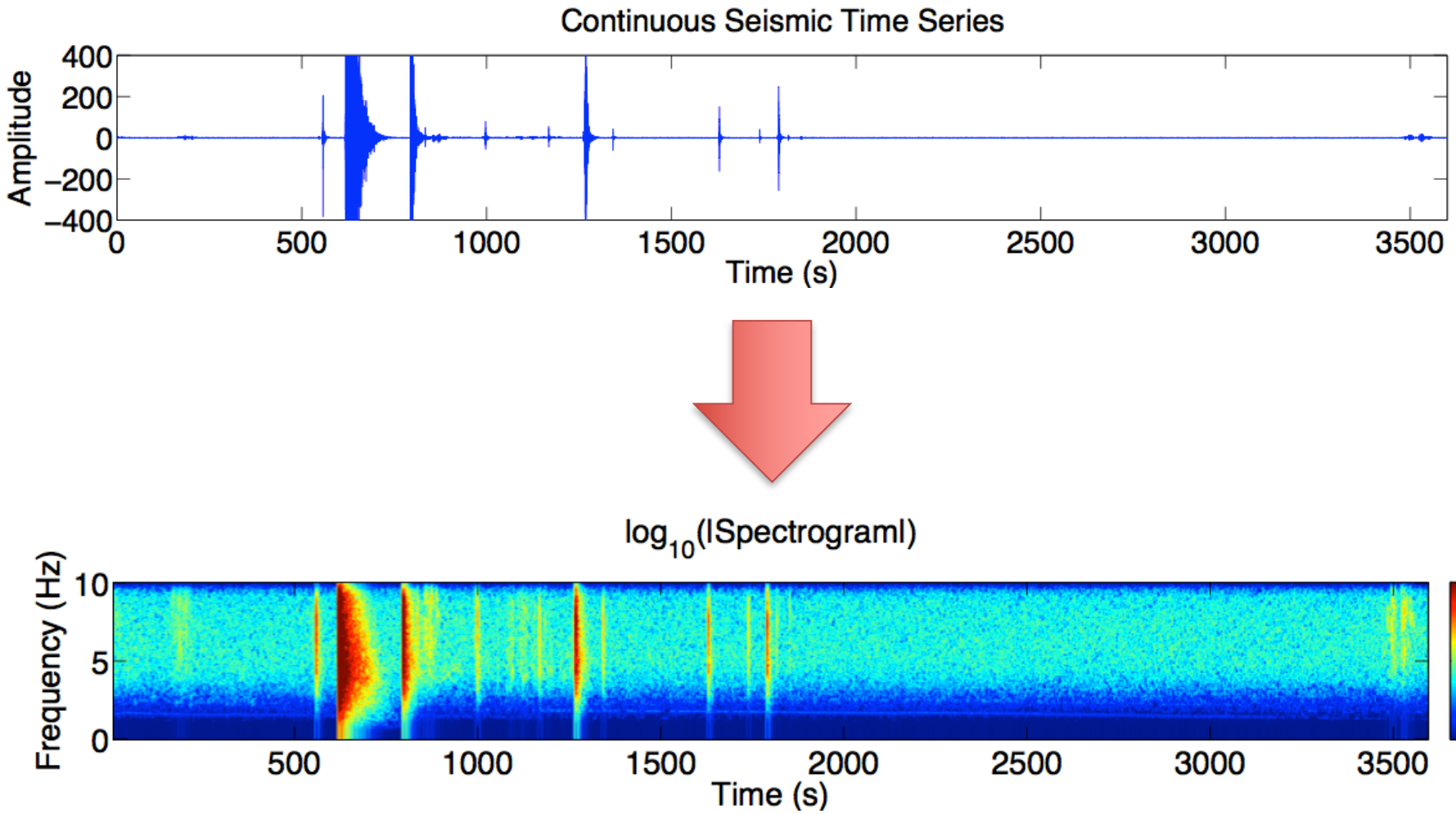
1. FAST Overview
2. Requirements and Installation
3. Hector Mine Tutorial
4. How to set parameters
    1. Input and Preprocessing
    2. Fingerprint
    3. Similarity Search
    4. Network Detection
    5. Output
5. References
6. Supplementary

Feature Extraction Overview



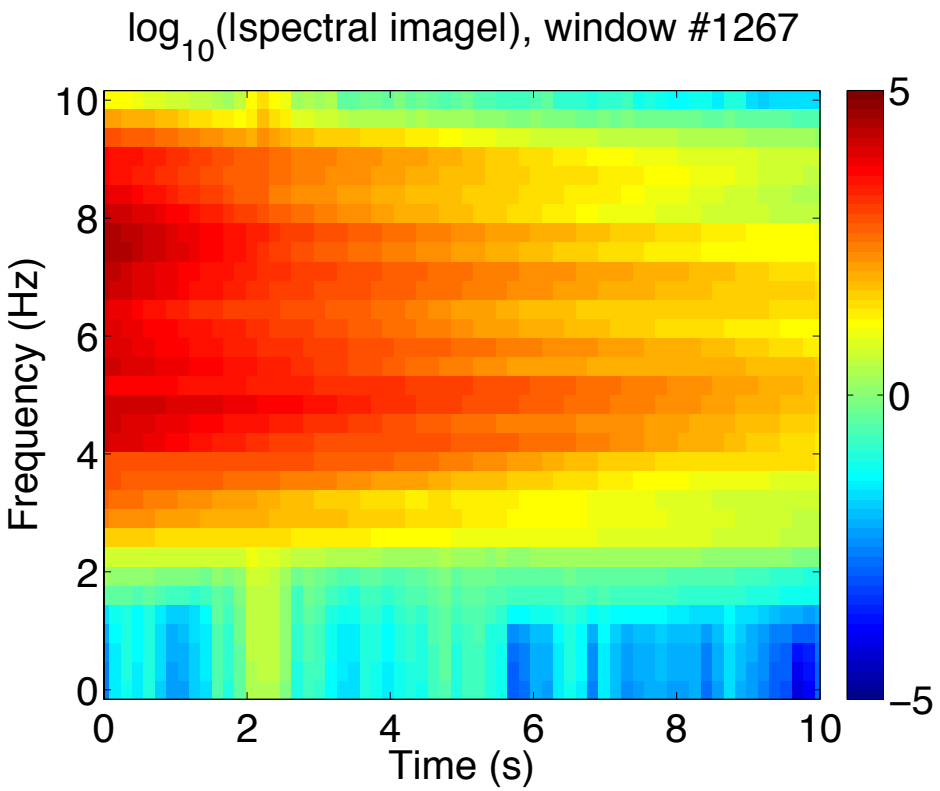


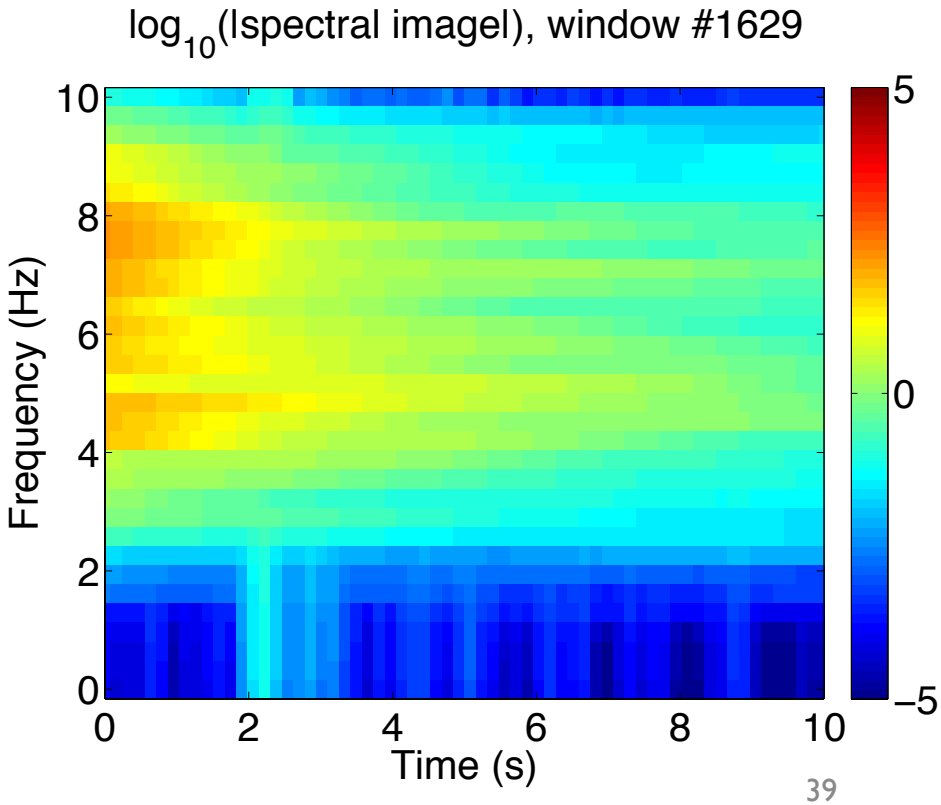
Step 1: Time Series ➔ Spectrogram



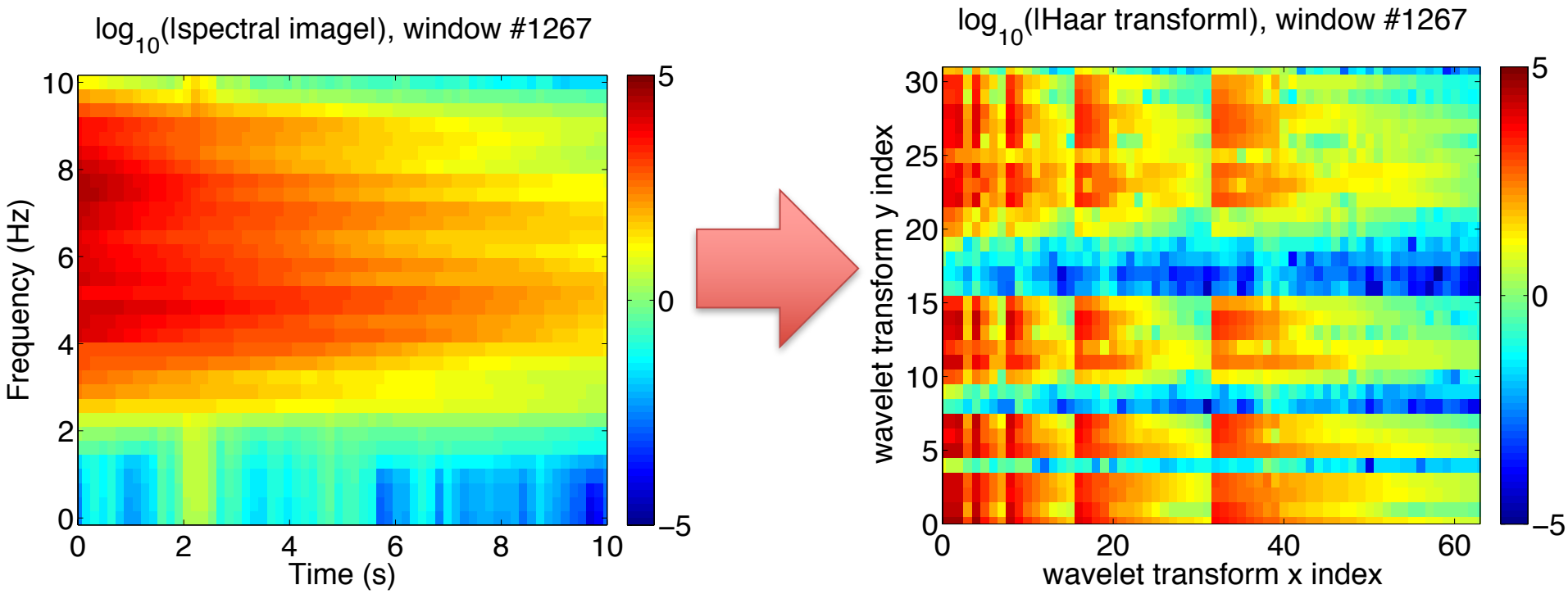
Step 2: Spectrogram ➔ Spectral Images

- To find short duration events, divide spectrogram into overlapping spectral images
  - Long lag ➔ fewer spectral images to compare ➔ fast



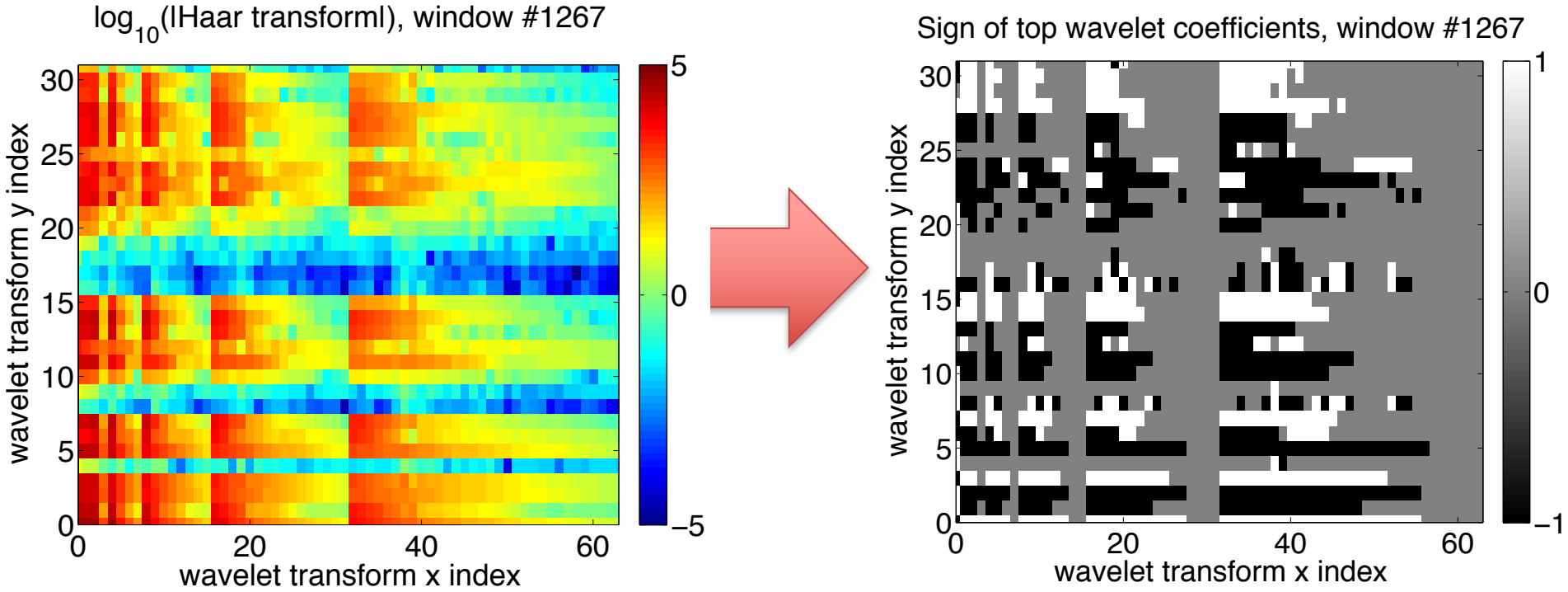


Step 3: Spectral Image ➔ Wavelet Transform



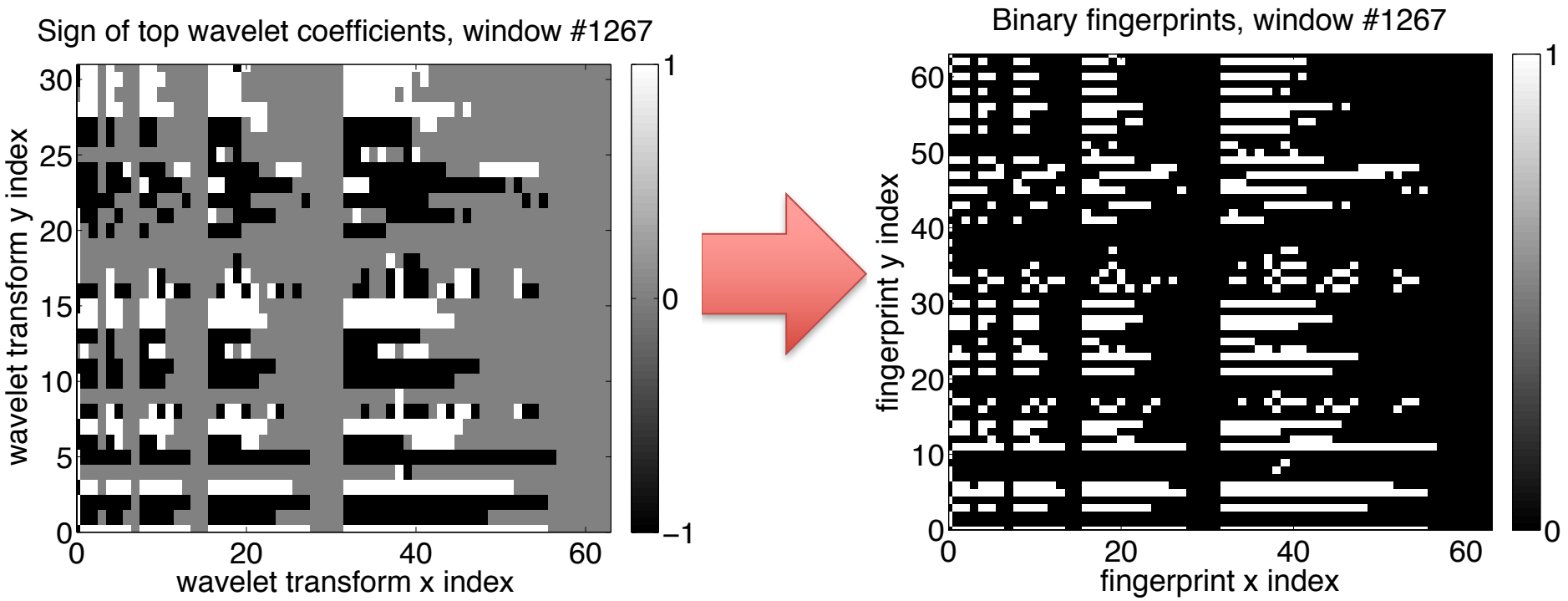
- Goal: compress nonstationary seismic signal
  - Compute 2D discrete wavelet transform (Haar basis) of spectral image to get wavelet coefficients

Step 4: Wavelet Transform ➔ Top Coefficients



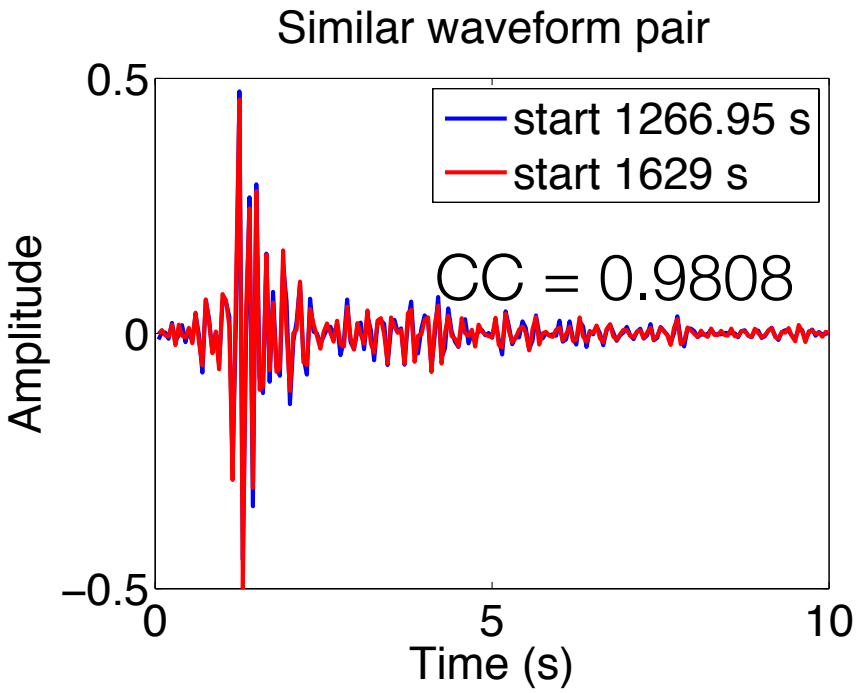
- Key discriminative features are concentrated in a few wavelet coefficients with highest deviation
  - Deviation defined by median/MAD over entire data set
  - Keep only sign (+ or -) of these coefficients, set rest to 0
- Data compression, robust to noise

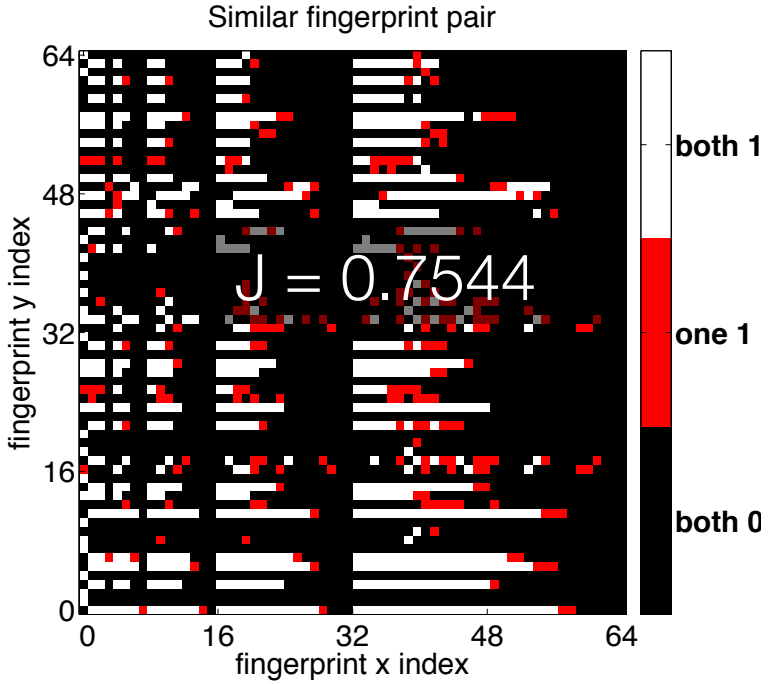
Step 5: Top Coefficients ➔ Binary Fingerprint



- Fingerprint must be compact and sparse to store in database
  - Convert top coefficients to a binary sequence of 0's, 1's
    - Negative: 01, Zero: 00, Positive: 10

How do we measure similarity?





Correlation Coefficient

$$CC(a,b) = \frac{\sum_{i=1}^{N} a_i b_i}{\sqrt{\sum_{i=1}^{N} a_i a_i}\sqrt{\sum_{i=1}^{N} b_i b_i}}$$

Jaccard Similarity

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{N_{white}}{N_{white} + N_{red}}$$

Fingerprint Parameters

```
{
   "fingerprint": {
      "sampling_rate": 20,
      "min_freq": 0.0,
      "max_freq": 10.0,
      "spec_length": 6.0,
      "spec_lag": 0.2,
      "fp_length": 32,
      "fp_lag": 5,
      "k_coef": 200,
      "nfreq": 32,
      "mad_sampling_rate": 1,
      "mad_sample_interval": 86400
   },
```

Sampling rate (Hz)
Bandpass frequency (Hz) - minimum
Bandpass frequency (Hz) – maximum
Time window length (s) for spectrogram
Time window lag (s) for spectrogram
Spectral image length (samples)
Spectral image lag (samples)
Number of wavelet coefficients to keep
Final spectral image width (samples)
Median/MAD sampling fraction of data
Median/MAD sampling frequency (s)

Need one input file per component at each station:
`parameters/fingerprint/`

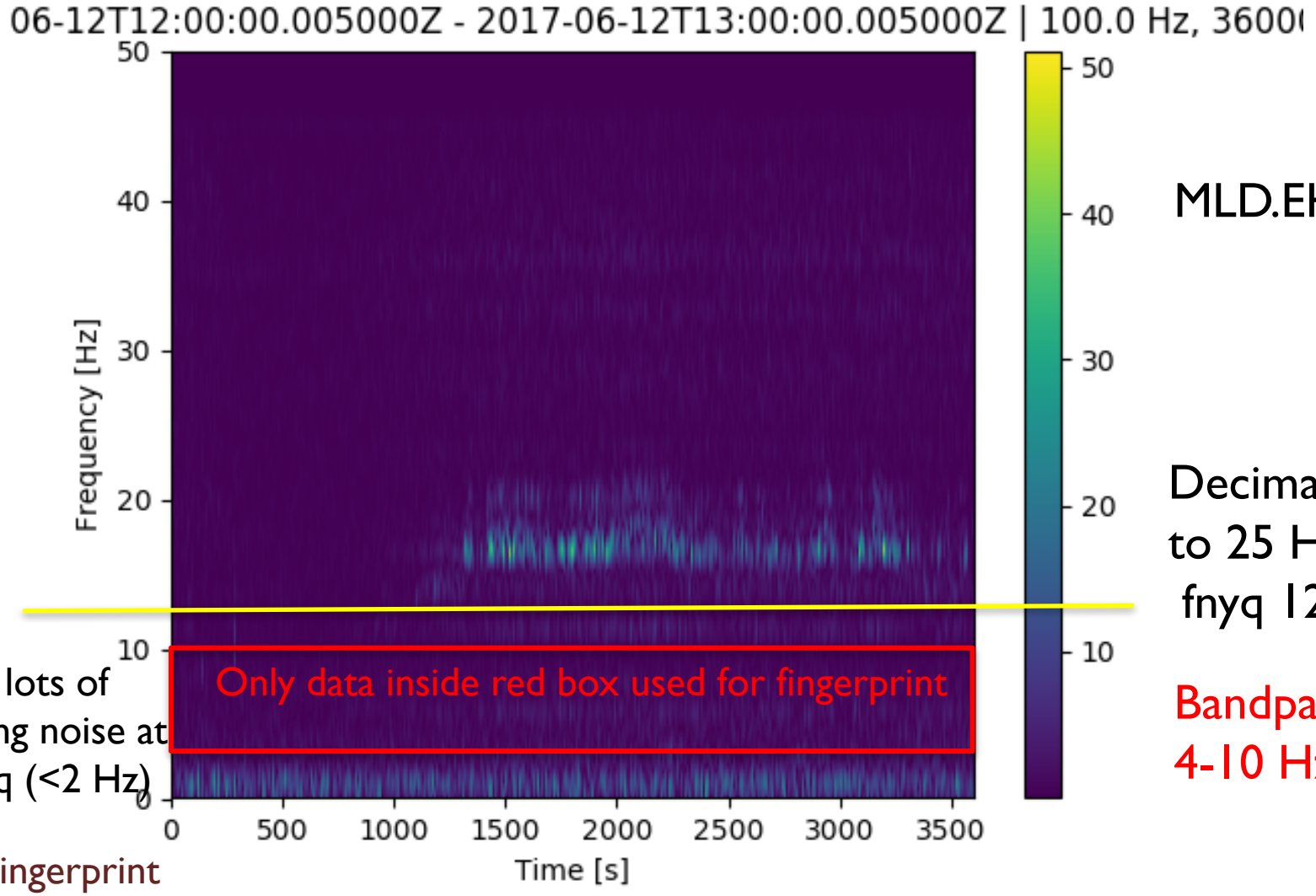
`fp_input_${NETWORK}_${STATION}_${CHANNEL}.json`

Example: `fp_input_CI_CDY_EHZ.json`

How to select bandpass filter?

- Filter can be different for different stations and components
- Contain as much of your desired earthquake signal as possible; not too narrowband
- Remove frequencies with repeated noise: important
  - View sample spectrograms to empirically determine these noisy frequencies (output as .png image files): `parameters/preprocess_utils/sample_spectrograms_daily_NEP.py`
    - Twice a day (day and night: cultural noise variations)
    - Once a month or once a day – sample randomly
  - Usually 0-2 Hz has repeated noise; sometimes >20 Hz
  - Without this step, similar noise signals will dominate your detections → you will not find earthquakes
- May want to avoid teleseismic event detection
  - Lower limit 3-4 Hz
- 4-12 Hz generally works well as default

Example: Bandpass filter selection, given sample spectrogram



MLD.EHN

Decimate 4x to 25 Hz: fnyq 12.5 Hz

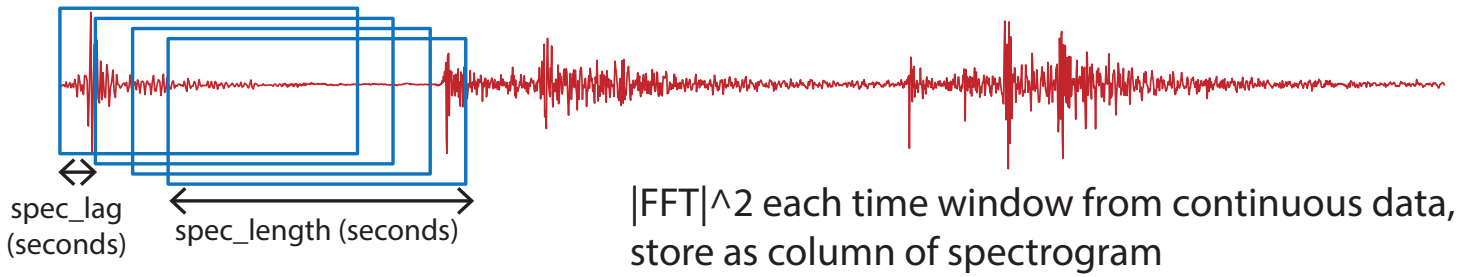
Bandpass 4-10 Hz

Usually lots of repeating noise at low freq (<2 Hz)

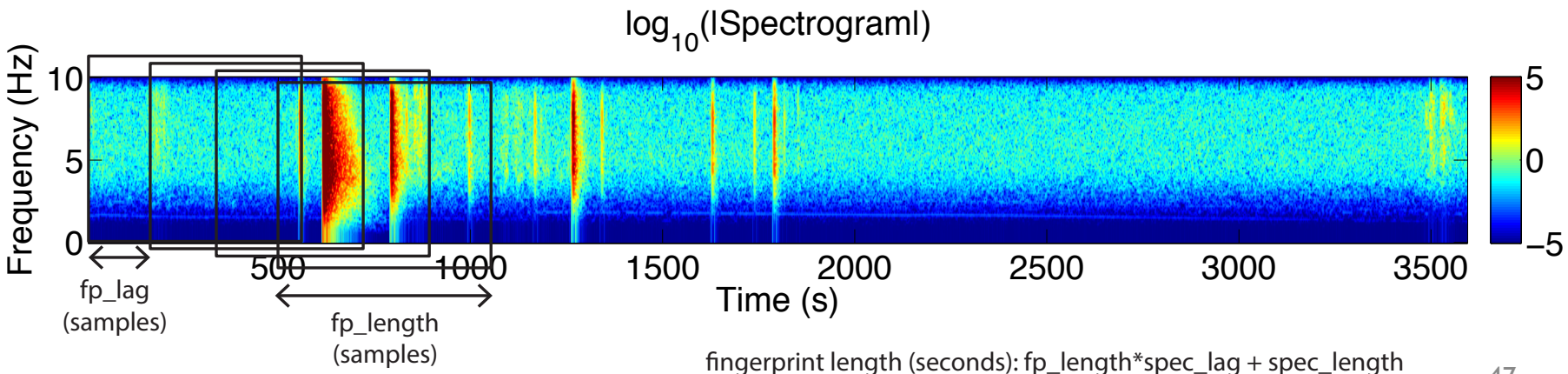
```
"spec_length": 6.0,
"spec_lag": 0.2,
"fp_length": 32,
"fp_lag": 5,
```

Time window length (s) for spectrogram
Time window lag (s) for spectrogram
Spectral image length (samples)
Spectral image lag (samples)

Spectral image (and fingerprint) length: 12.4 seconds
Spectral image (and fingerprint) lag: 1 second



|FFT|^2 each time window from continuous data, store as column of spectrogram



4.2 Fingerprint

"dt_fp"

fingerprint length (seconds): fp_length*spec_lag + spec_length
fingerprint lag (seconds): fp_lag*spec_lag

"spec_length": 6.0,	Time window length (s) for spectrogram
"spec_lag": 0.2,	Time window lag (s) for spectrogram
"fp_length": 32,	Spectral image length (samples)
"fp_lag": 5,	Spectral image lag (samples)

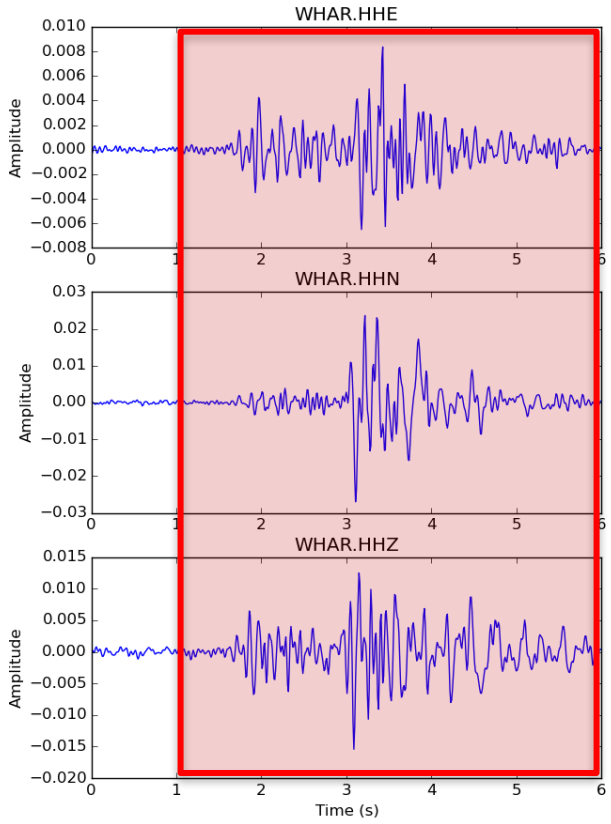
Spectral image (and fingerprint) length: 12.4 seconds
Spectral image (and fingerprint) lag: 1 second

- Choose parameters so that entire earthquake waveform (P,S,coda) fits into fp_length time window
  - Not too short (just P or S arrival), but not too long (otherwise adds noise)
  - Choose same parameters for all components at all stations
  - Adjust spec_length and fp_length values
  - fp_length (samples) should be a power of 2
    - Required for the wavelet transform. If fp_length is not a power of 2, each spectral image will be downsampled to the next smallest power of 2.

- spec_lag should be short with >95% overlap between adjacent time windows; 0.05 to 0.2 seconds is a good default
- fp_lag: can be slightly longer with >85% overlap between adjacent spectral images; 0.5 to 2 seconds is a good default

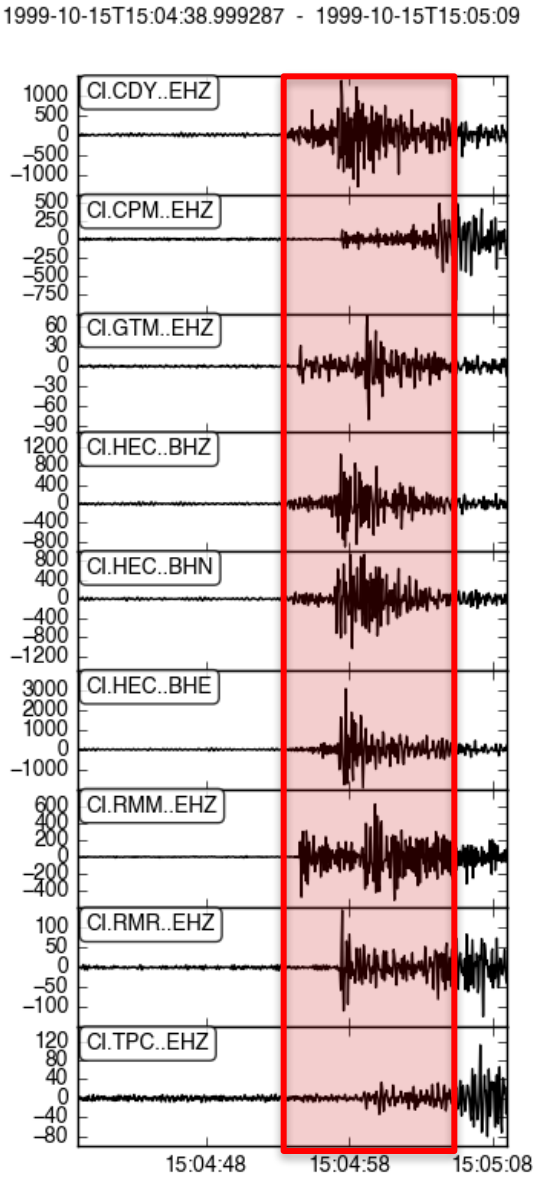
Guy, Arkansas

Spectral image length:
4.92 seconds



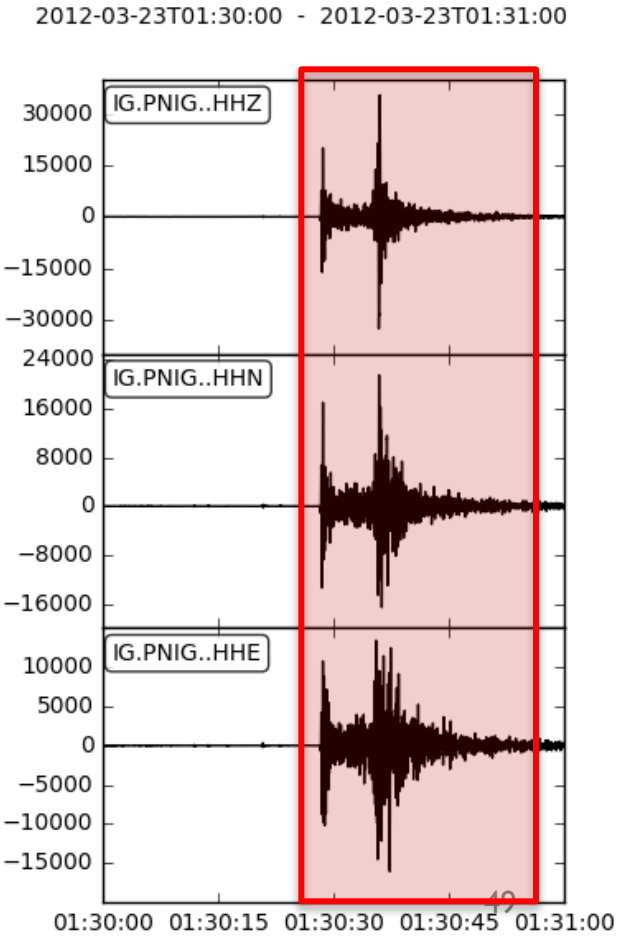
Hector Mine, CA

Spectral image length:
12.4 seconds



Ometepec, Mexico

Spectral image length:
31.6 seconds



4.2 Fingerprint

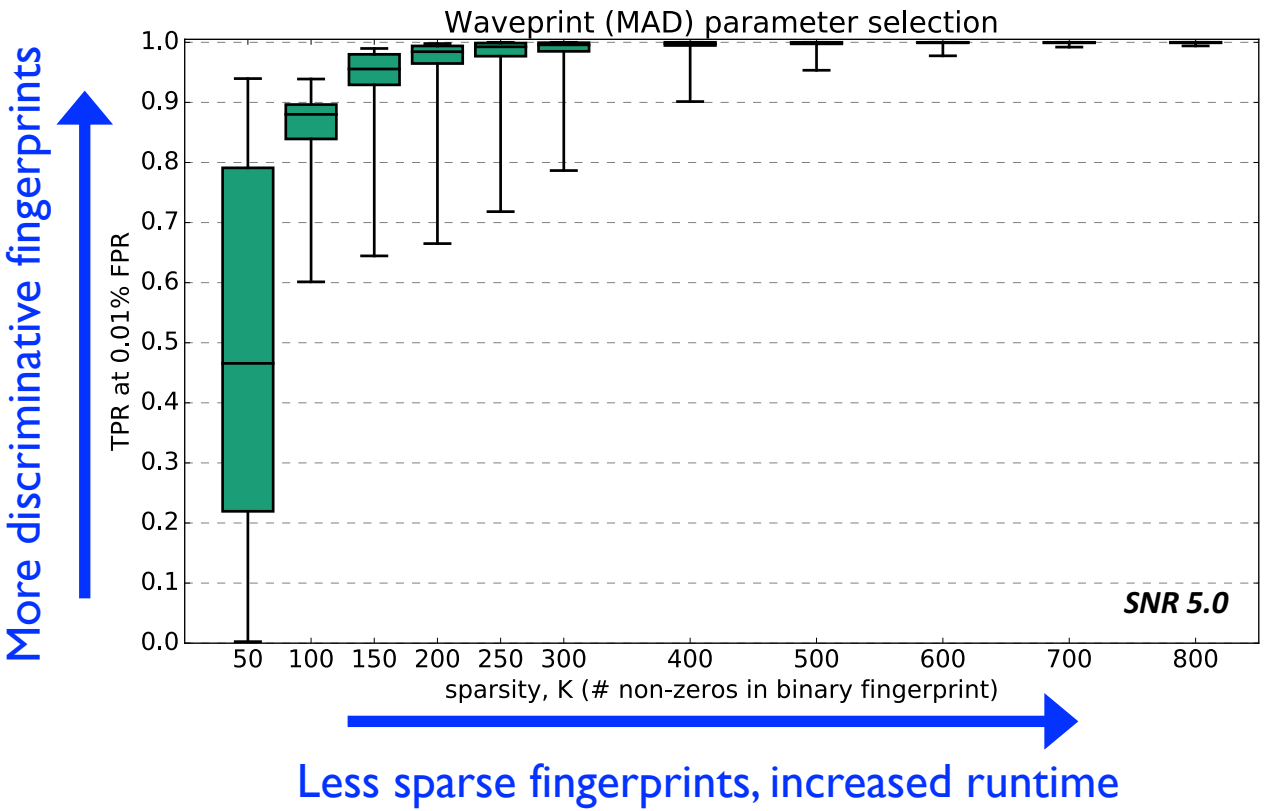
"k_coef": 200, Number of wavelet coefficients to keep
"nfreq": 32, Final spectral image width (samples)

- Spectral image width will be resized (usually downsampled) to nfreq samples
  - Initial spectral image width is 2*(spec_length)*(max_freq - min_freq) samples: depends on bandpass filter
  - nfreq must be a power of 2 for wavelet transform; 32 is good default value
- Each spectral image (and wavelet-transform) has dimensions = nfreq * fp_length
  - This example: 32*32 = 1024 elements
- To set k_coef, keep ~20% of most anomalous wavelet coefficients
  - This example: 200/1024 ~ 19.5%
- Fingerprint has twice as many elements as spectral image
  - This example: 2 * 1024 = 2048 elements

"k_coef": 200,

Number of wavelet coefficients to keep

FAST Fingerprints: Sparsity Parameter



For binary fingerprints of dimension 4096 with K non-zeros, fingerprint: the optimal value of K is in the range of 300-500 (about 10% sparsity).

Vertical is a measure of how well the fingerprints separate signal from noise varies with sparsity. The sparsity is on the horizontal axis - less sparse fingerprints (larger K) tend to have longer similarity search runtimes. Box plots show variation in performance over 8 data sets.

(Bergen & Beroza, 2018b - *in preparation)*

"mad_sampling_rate": 1, Median/MAD sampling fraction of data
"mad_sample_interval": 86400 Median/MAD sampling interval (s)

- For each coefficient, compute median/MAD statistics over entire data set. This step determines which k_coef wavelet coefficients to keep.
  - But if data set is too long (months-years), compute statistics over a representative sample of data
- mad_sampling_rate: fraction of entire continuous data set used to compute median/MAD
  - Duration of sampled data set = (mad_sampling_rate) * (continuous data duration) should not exceed 1 week, otherwise may not fit in memory
- Retrieve sample of data for median/MAD statistics once per mad_sample_interval
  - 86400 s = 1 day is a good default value
  - Exact time of sample is determined randomly: expect different numbers every time you run this
- Median/MAD for each coefficient: pre-computed and stored in a text file, then read in during fingerprint generation
  - `data/waveforms${STATION}/mad/mad*.txt`

"mad_sampling_rate": 1, Median/MAD sampling fraction of data
"mad_sample_interval": 86400 Median/MAD sampling interval (s)

- Rule of thumb
  - Duration <1 week: set to 1 (use entire data set for sample)
  - Duration weeks – months: set to 0.1 (sample 10% of data)
  - Duration >1 year: set to 0.01 (sample 1% of data)
- Runtime-accuracy tradeoff

Sampling Rate	Average Fingerprint Overlap	Speedup
0.001	94.9%	350x
0.01	98.7%	99.8x
0.1	99.5%	10.5x
0.5	99.7%	2.2x
0.9	99.9%	1.1x

Table 3: Speedup and quality of different MAD sampling rate compared to no sampling on 1.3M fingerprints. Sampling enables a 100x speed up in MAD calculation while 98.7% accuracy. Below 1%, runtime improvements suffer from a diminishing return, as the IO begins to dominate the MAD calculation in runtime.

```
"performance": {
    "num_fp_thread": 8,
    "partition_len": 86400
},
```

Number of parallel processes
Continuous data partition (s)

- Can generate fingerprints in parallel by setting num_fp_thread > 1
  - Each process gets one continuous data file
    - Better parallelization if you have one mseed file per day, rather than one per month
- Generate fingerprints partition_len at a time
  - 86400 s (1 day) is a good default value
- Changing these "performance" parameters should not affect the final results

```
"data": {
    "station": "CDY",
    "channel": "EHZ",
    "start_time": "99-10-15T13:00:00.0",
    "end_time": "99-10-16T09:46:44.0",
    "folder": "../data/waveformsCDY/",
    "fingerprint_files": [
      "Deci5.Pick.19991015130000.CI.CDY.EHZ.sac"],
    "MAD_sample_files": [
      "Deci5.Pick.19991015130000.CI.CDY.EHZ.sac"]
  }
```

Time format: YY-MM-DDTHH:MM:SS.S

folder with input data

Usually fingerprint_files, MAD_sample_files should be the same; can have list of multiple files for input continuous data

- OUTPUTS
  - ${folder}/fingerprints/
    - Fingerprints from each continuous data file (can delete these later): `fp_*`
    - Single file with all fingerprints: `${STATION}.${CHANNEL}.fp`
    - For example: `data/waveformsCDY/fingerprints/CDY.EHZ.fp`
  - ${folder}/timestamps/
    - Timestamp (YYYY-MM-DDTHH:MM:SS.SSS) for each fingerprint
    - Timestamps from each continuous data file (used later for global index calculation): `ts_*`

Global Index: Inputs & Outputs

- Input file: `global_indices.json`

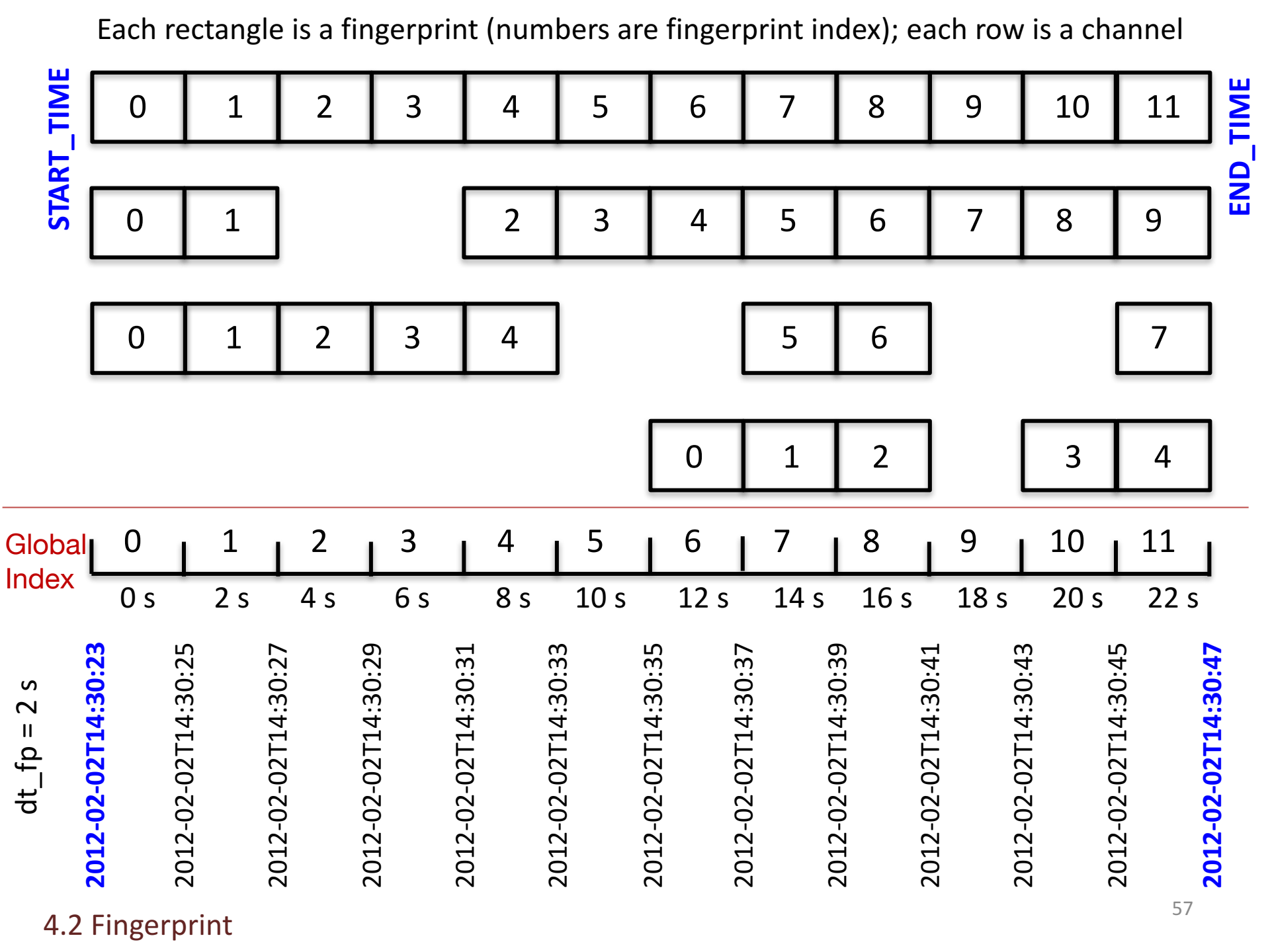
```
{
    "index_folder": "../data/global_indices/",
    "fp_param_dir": "../parameters/fingerprint/",
    "fp_params": ["fp_input_CI_TPC_EHZ.json", "fp_input_CI_RMR_EHZ.json",
    "fp_input_CI_RMM_EHZ.json", "fp_input_CI_HEC_BHE.json",
    "fp_input_CI_HEC_BHN.json", "fp_input_CI_HEC_BHZ.json",
    "fp_input_CI_CPM_EHZ.json", "fp_input_CI_GTM_EHZ.json",
    "fp_input_CI_CDY_EHZ.json"]
}
```

Output folder

Input folder

Fingerprint input files for all components and stations to use for detection

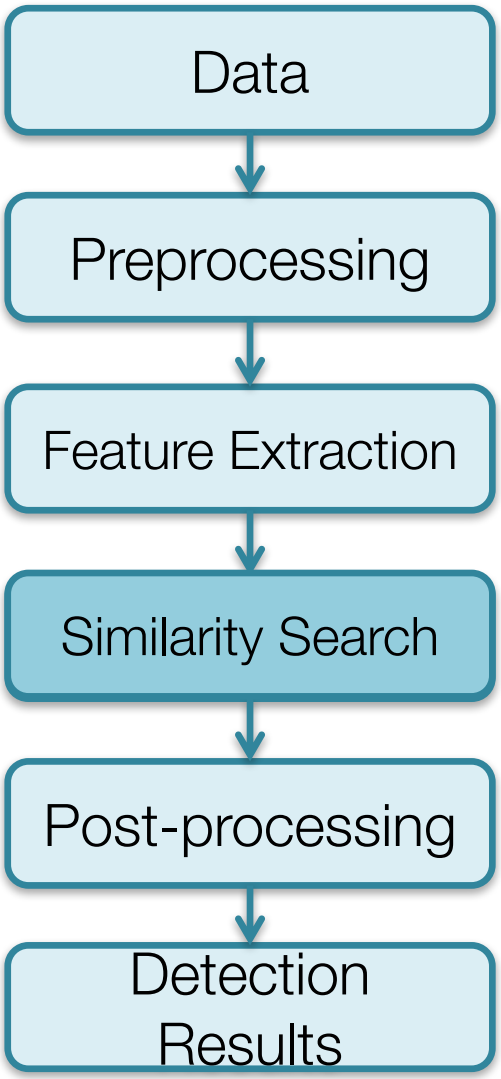
- Outputs:
  - Global start time "t0" for index 0 (YYYY-MM-DD THH:MM:SS.SSS), fingerprint input file names: `global_idx_stats.txt`
  - Global index files for each channel, containing global index of each fingerprint: `${STATION}_${CHANNEL}_idx_mapping.txt`


Each rectangle is a fingerprint (numbers are fingerprint index); each row is a channel
START_TIME
END_TIME
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7
0 1 2 3 4
Global Index
0 1 2 3 4 5 6 7 8 9 10 11
0 s 2 s 4 s 6 s 8 s 10 s 12 s 14 s 16 s 18 s 20 s 22 s
dt_fp = 2 s
2012-02-02T14:30:23
2012-02-02T14:30:25
2012-02-02T14:30:27
2012-02-02T14:30:29
2012-02-02T14:30:31
2012-02-02T14:30:33
2012-02-02T14:30:35
2012-02-02T14:30:37
2012-02-02T14:30:39
2012-02-02T14:30:41
2012-02-02T14:30:43
2012-02-02T14:30:45
2012-02-02T14:30:47

Outline

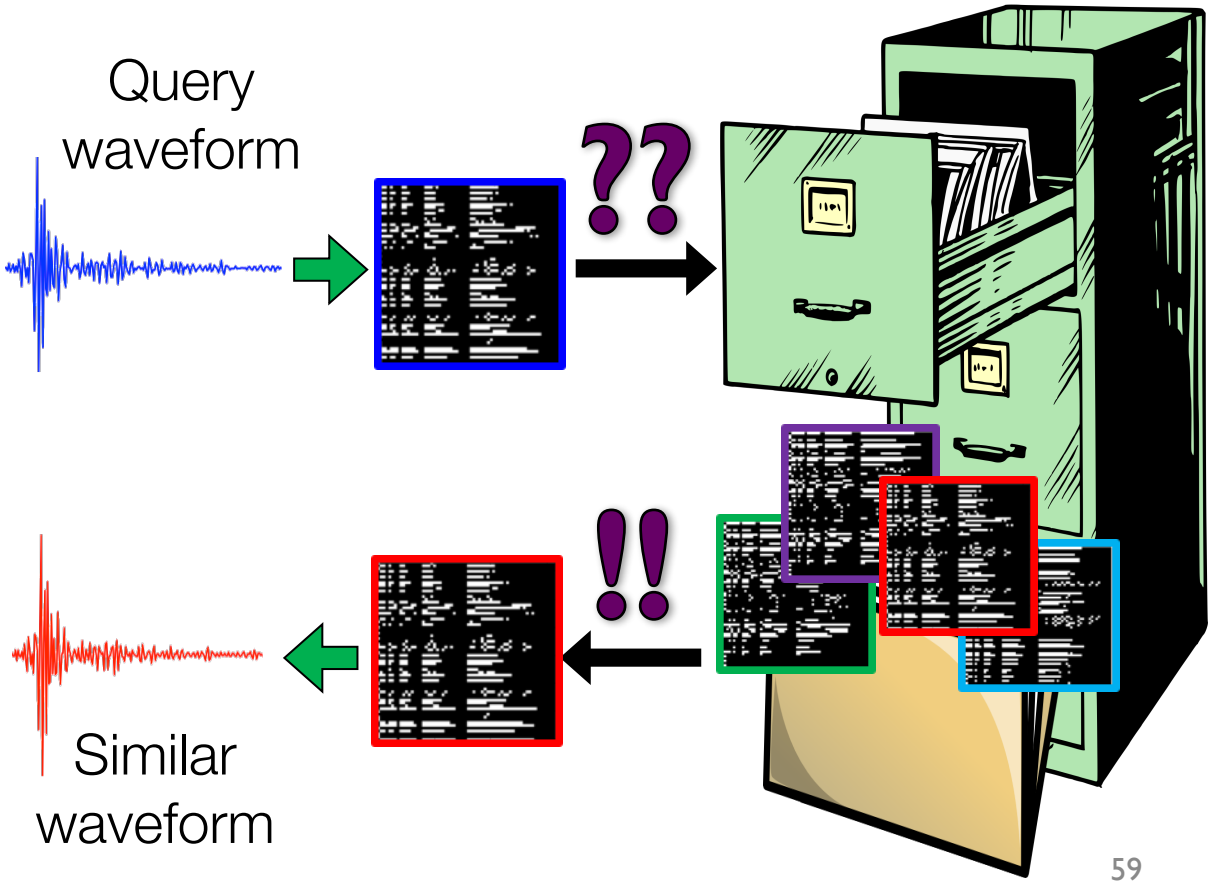
1. FAST Overview
2. Requirements and Installation
3. Hector Mine Tutorial
4. How to set parameters
    1. Input and Preprocessing
    2. Fingerprint
    3. Similarity Search
    4. Network Detection
    5. Output
5. References
6. Supplementary

Similarity Search Overview



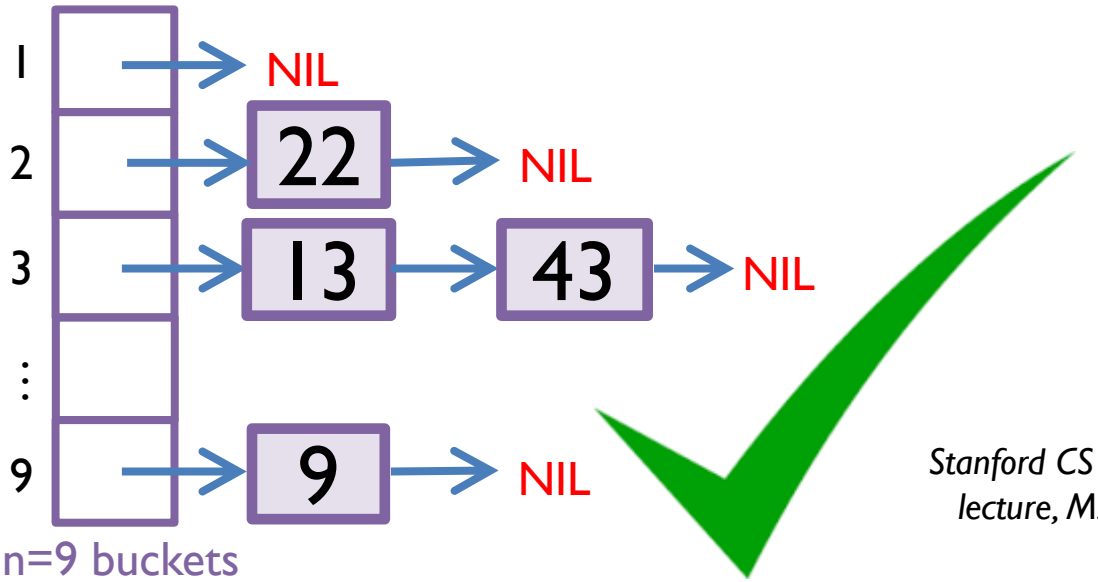
Fast approximate similarity search

- MinHash (probabilistic estimate of Jaccard similarity)
- Locality Sensitive Hashing (LSH)



Hashing

Efficient insertion, search, removal of items in databases



Stanford CS 161 course lecture, M. Wootters

Min-Hash*

Fingerprint

$h_i(x)$

Min-Hash function
(choose randomly)

155

Integer

Repeat i = 1, …, p times to get array of integers:
Min-Hash Signature (MHS)

A

B

$$\Pr\left[h(A) = h(B)\right] = J(A,B)$$

Min-Hash Signature similarity

Jaccard similarity

Broder et al. (2000)

Reduce fingerprint dimensionality, preserve similarity

Example: Min-Hash Signature (MHS)

$$\Pr\left[h(A) = h(B)\right] = J(A,B)$$

5/6 = 0.8333 0.7544

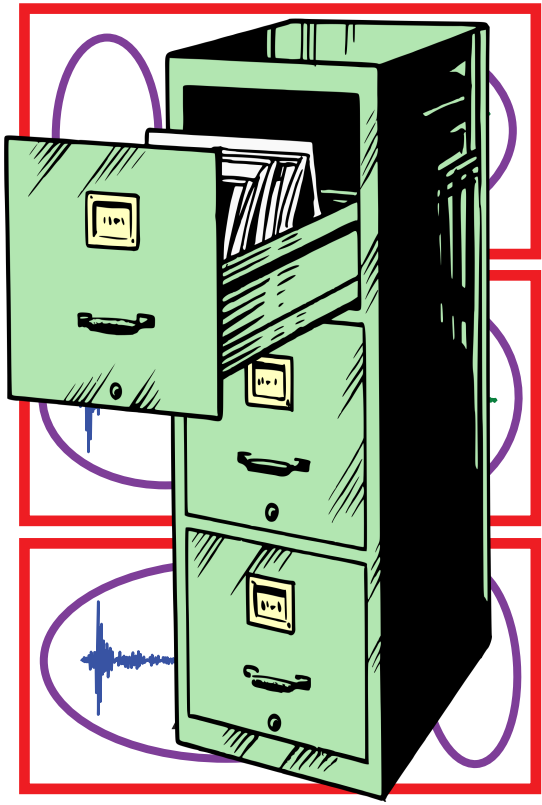
A	$h_i(x)$	integer	B	$h_i(x)$	integer	
	→	155		→	155	✔
	→	64		→	64	✔
	→	231		→	207	✗
	→	35		→	35	✔
	→	110		→	110	✔
	→	21		→	21	✔

LSH* Example: Constructing Database

Hash buckets:

	A	h(A)	B	h(B)	MHS subset match?
Table 1		155		155	Yes
		64		64	
Table 2		231		207	No
		35		35	
Table 3		110		110	Yes
		21		21	

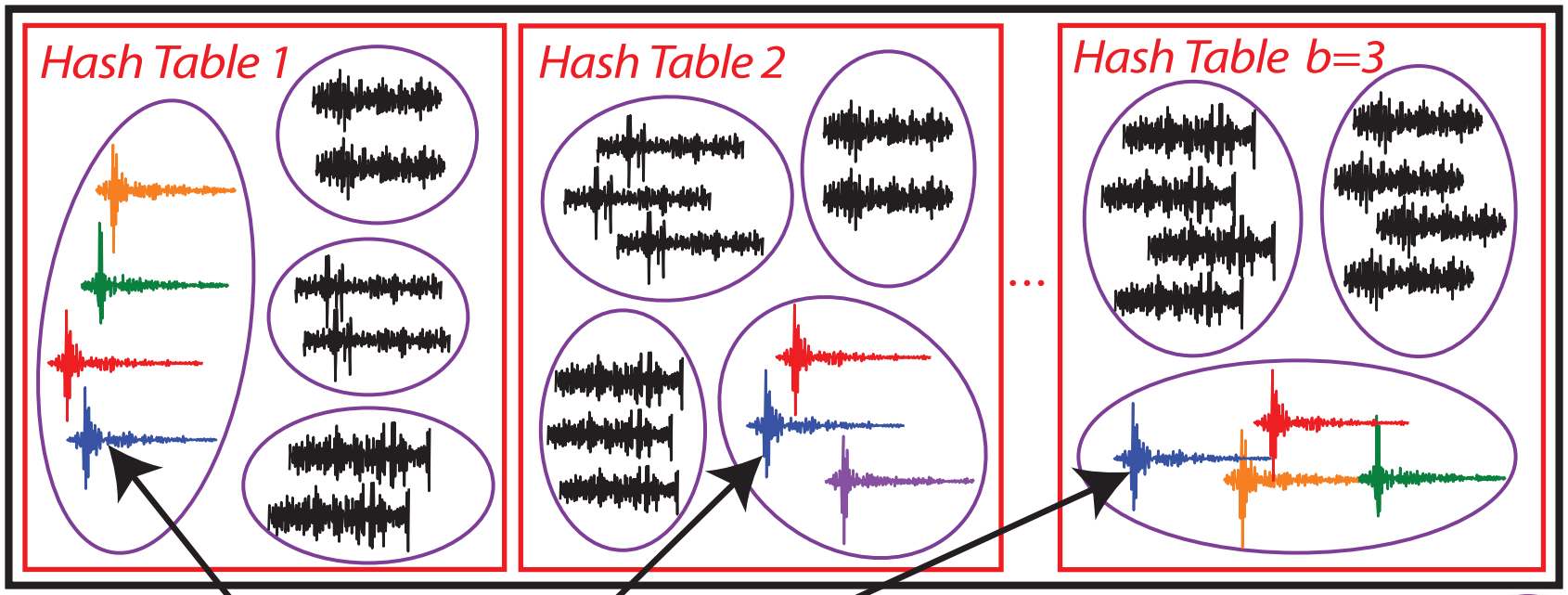
Database



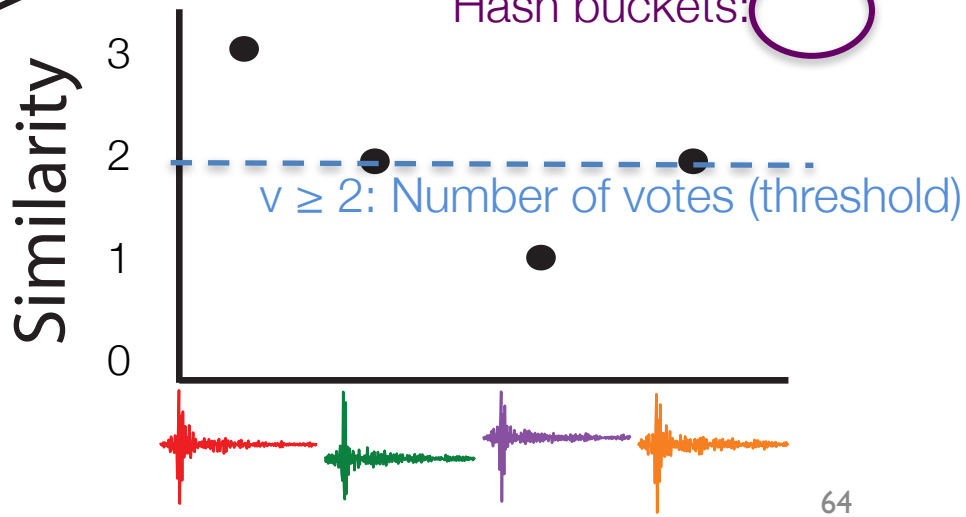
p=6 items in Min-Hash Signature (MHS) =
(r = 2 hash functions per table)*(b = 3 hash tables)

*Andoni and Indyk (2008)

Similarity Search in Fingerprint Database

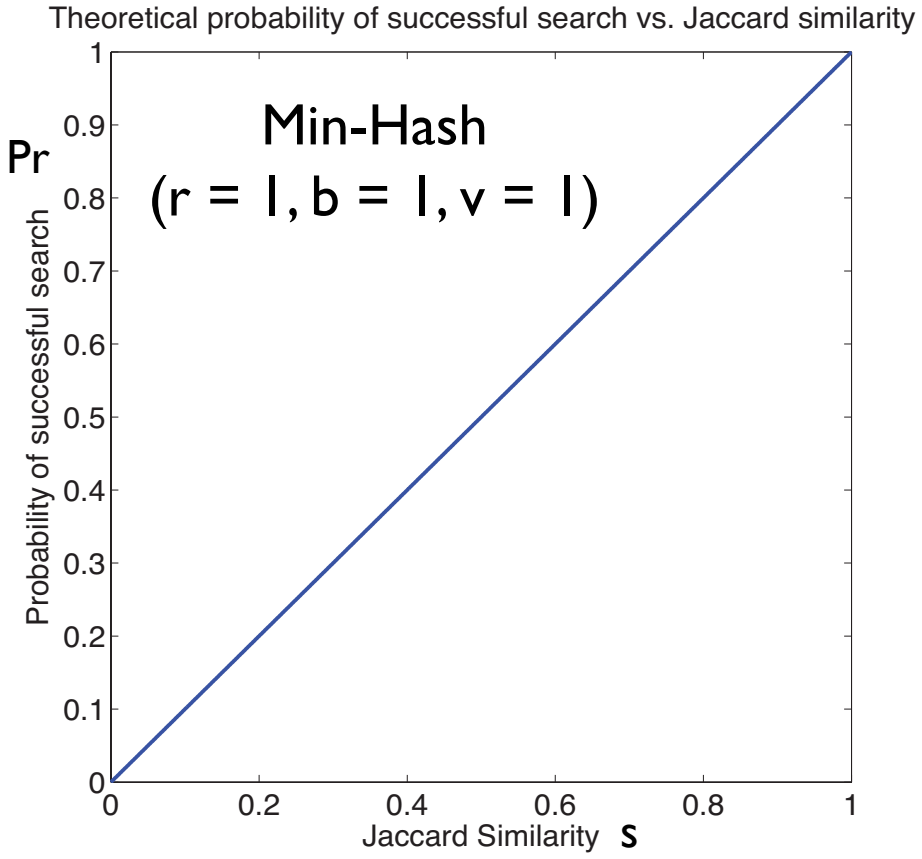


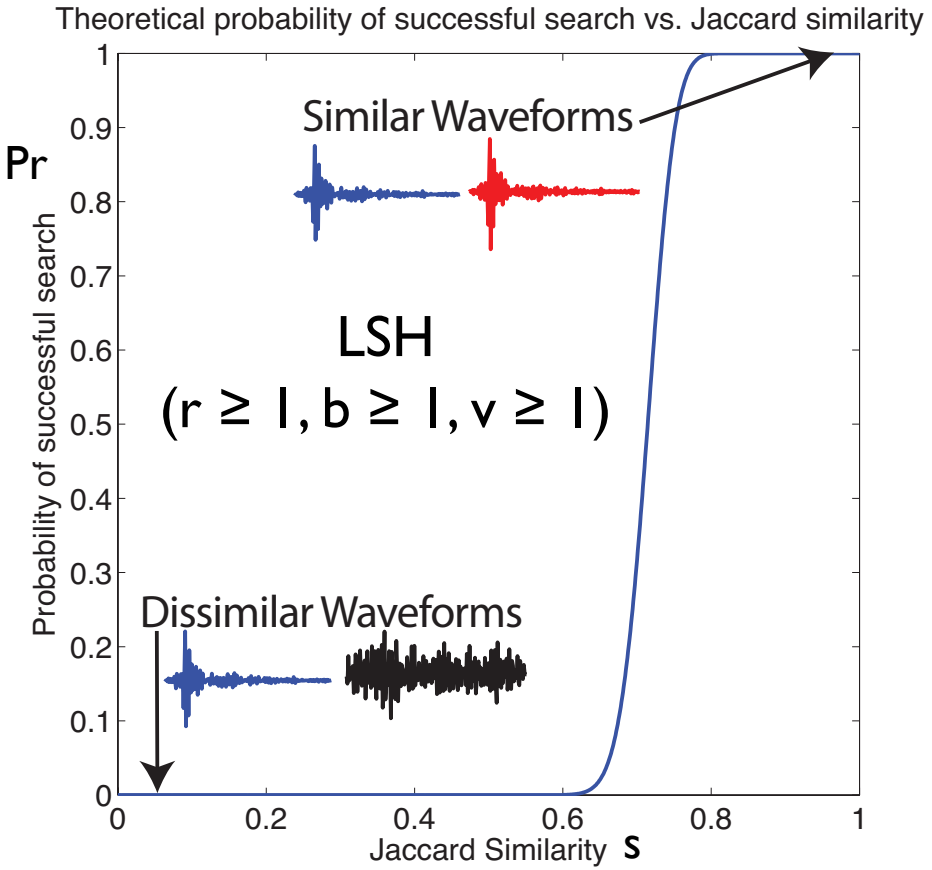
Hash buckets:



Similarity: Number of hash tables with fingerprint pair in same hash bucket

Probability of Detection





$$\Pr\left[h(A) = h(B)\right] = J(A,B)$$

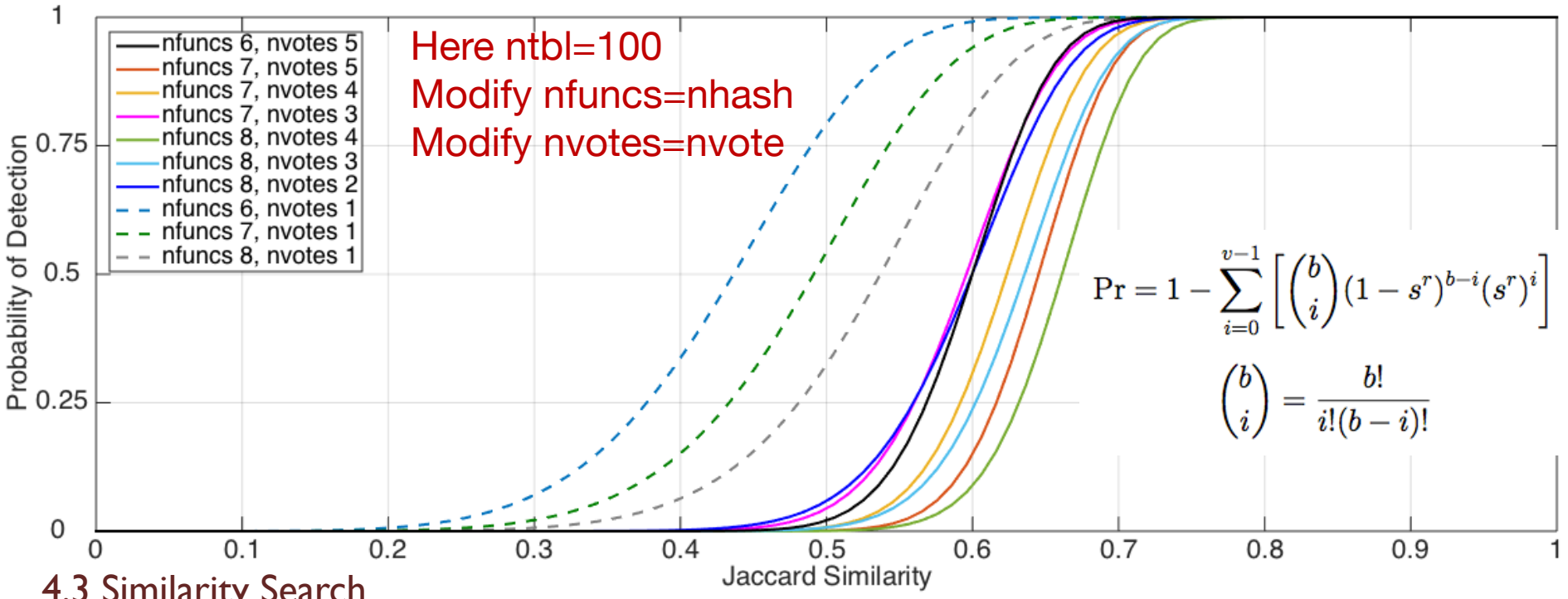
Pr = s

$$\Pr = 1 - \sum_{i=0}^{v-1} \left[\binom{b}{i} (1 - s^r)^{b-i} (s^r)^i \right]$$

$$\binom{b}{i} = \frac{b!}{i!(b-i)!}$$

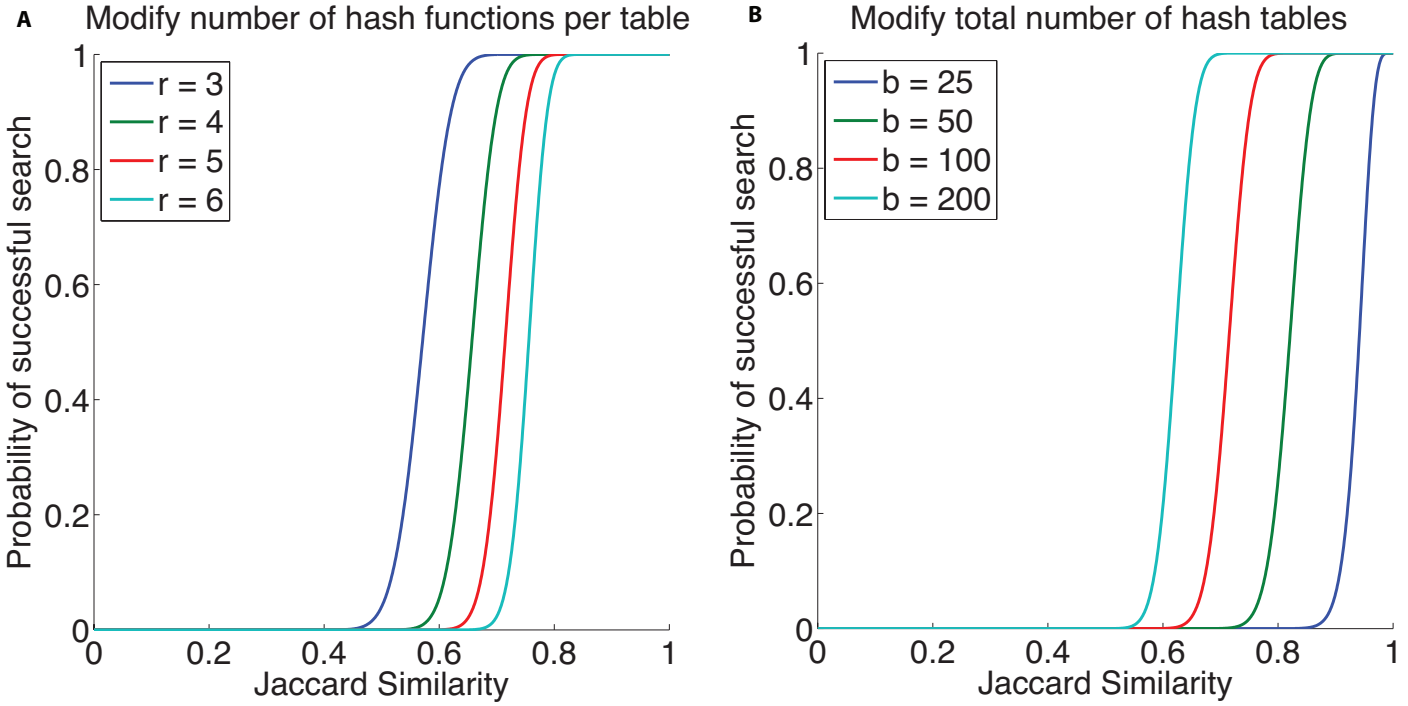
```
"lsh_param": {
        "ntbl": 100,
        "nhash": 4,
        "nvote": 2,
        "nthread": 8,
        "npart": 1,
        "repeat": 5
    },
```

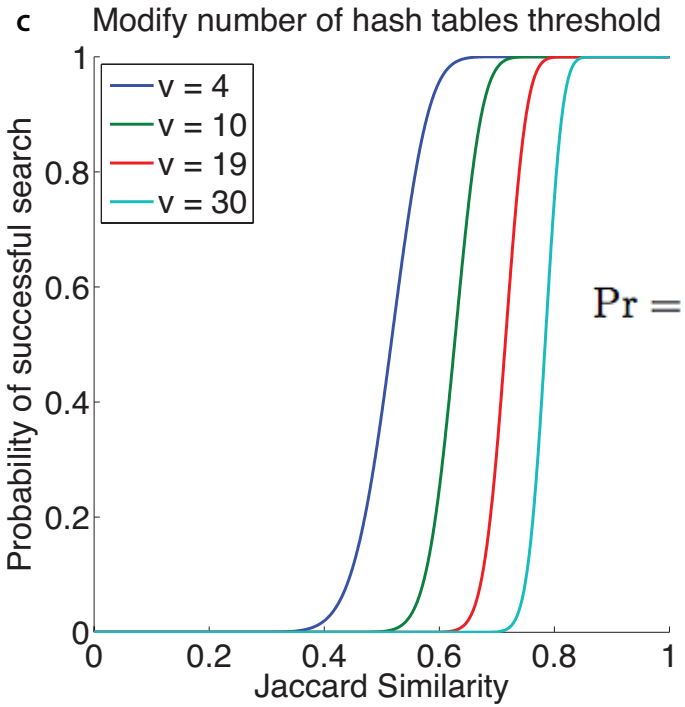
```
{
    b: Number of hash tables
    r: Number of hash functions per table
    v: Number of votes
    Number of threads for parallel processing
    Number of partitions for the database
    Near-repeat exclusion parameter (samples)
},
```



LSH Parameter Guidance

- ntbl (b): Number of hash tables
  - 100 is good default value
- nhash (r): Number of hash functions per table
  - Most sensitive parameter; significant effect on detection performance
  - Lower values: fewer missed detections, more false detections, longer runtime
  - Higher values: more missed detections, fewer false detections, shorter runtime
  - Suggested values (only possibilities are 1, 2, 3, 4, 5, 6, 7, 8):
    - nhash=4 for shorter duration data sets (days – weeks)
    - nhash=5 for longer data sets (months – year)
    - nhash=6 for longest data sets (5-10 years)
- nvote (v): Number of votes (pair of similar fingerprints must be in same hash bucket in at least v out of b hash tables)
  - Can use nvote as threshold for single station detection
  - nvote=2 is good starting value; initially set low, can increase threshold later during network detection
- repeat: Near-repeat exclusion parameter
  - Avoid detecting any fingerprint with itself (or slight offset to itself), which is guaranteed to be similar
  - 5 samples is good default value (Multiply by dt_fp to get value in seconds)



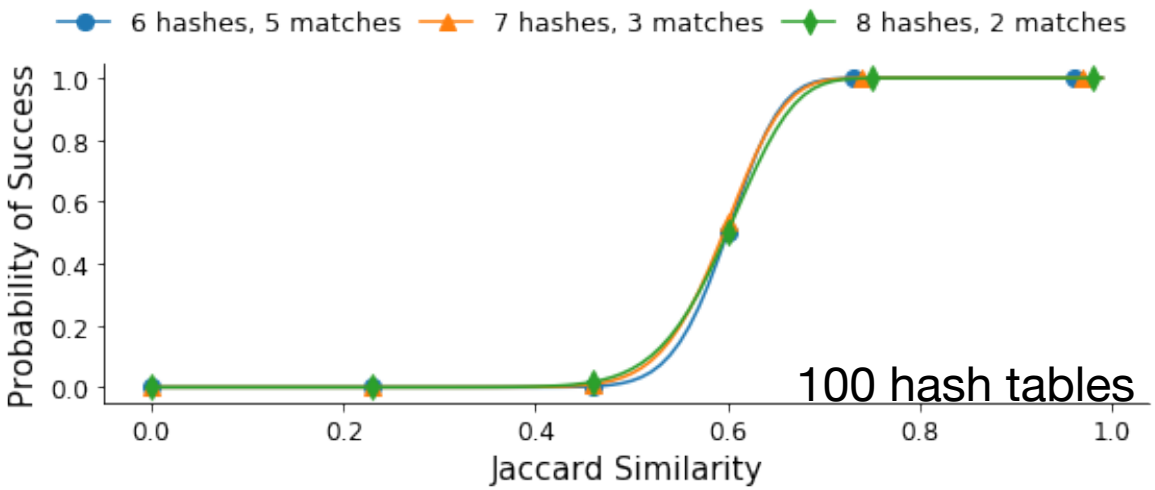


The Jaccard similarity threshold (fast increasing part of the S-curve) increases with the increase of number of hash functions (r), number of votes (v) and the decrease of number of tables (b)

$$\Pr = 1 - \sum_{i=0}^{v-1} \left[\binom{b}{i} (1 - s^r)^{b-i} (s^r)^i \right]$$

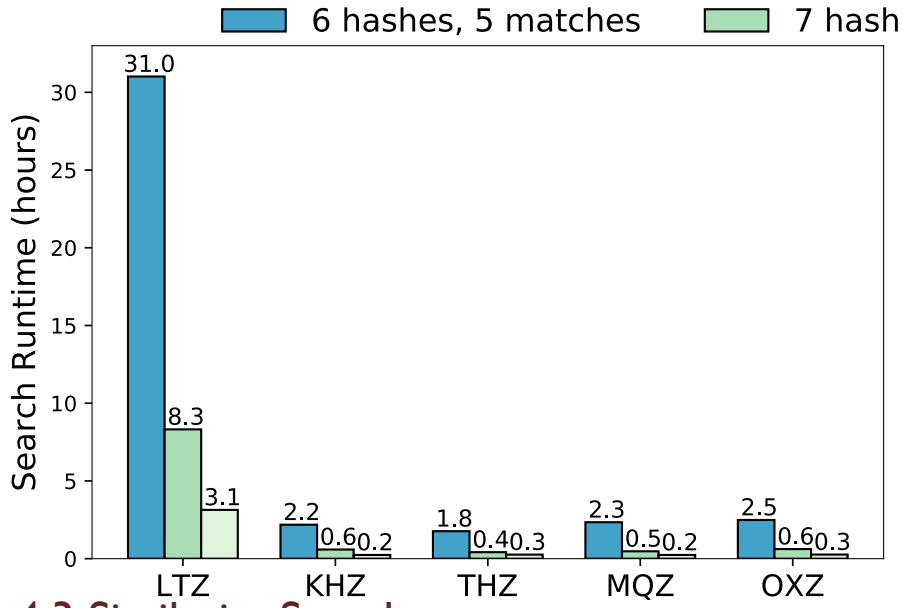
$$\binom{b}{i} = \frac{b!}{i!(b-i)!}$$

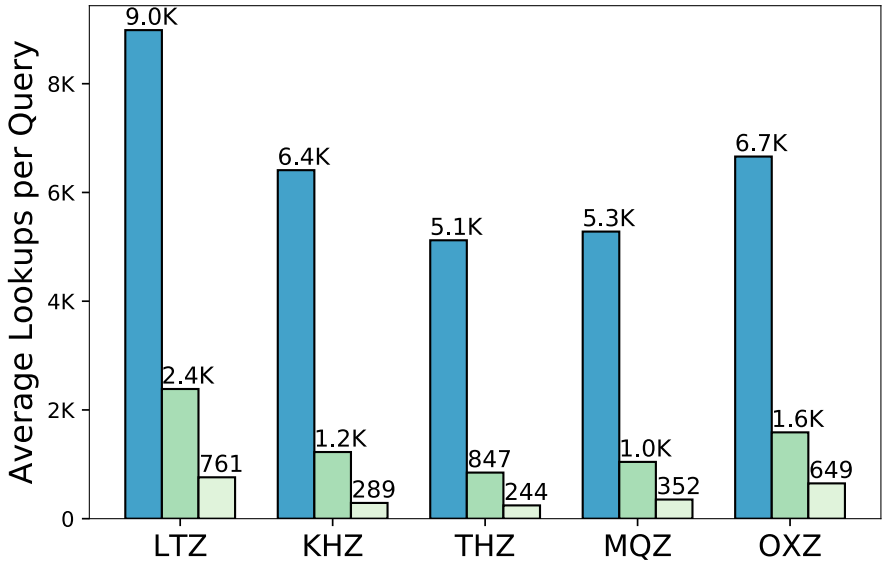
Performance Impact of LSH parameters



The three parameter settings have near identical theoretical detection probability, but up to 10x difference in runtime

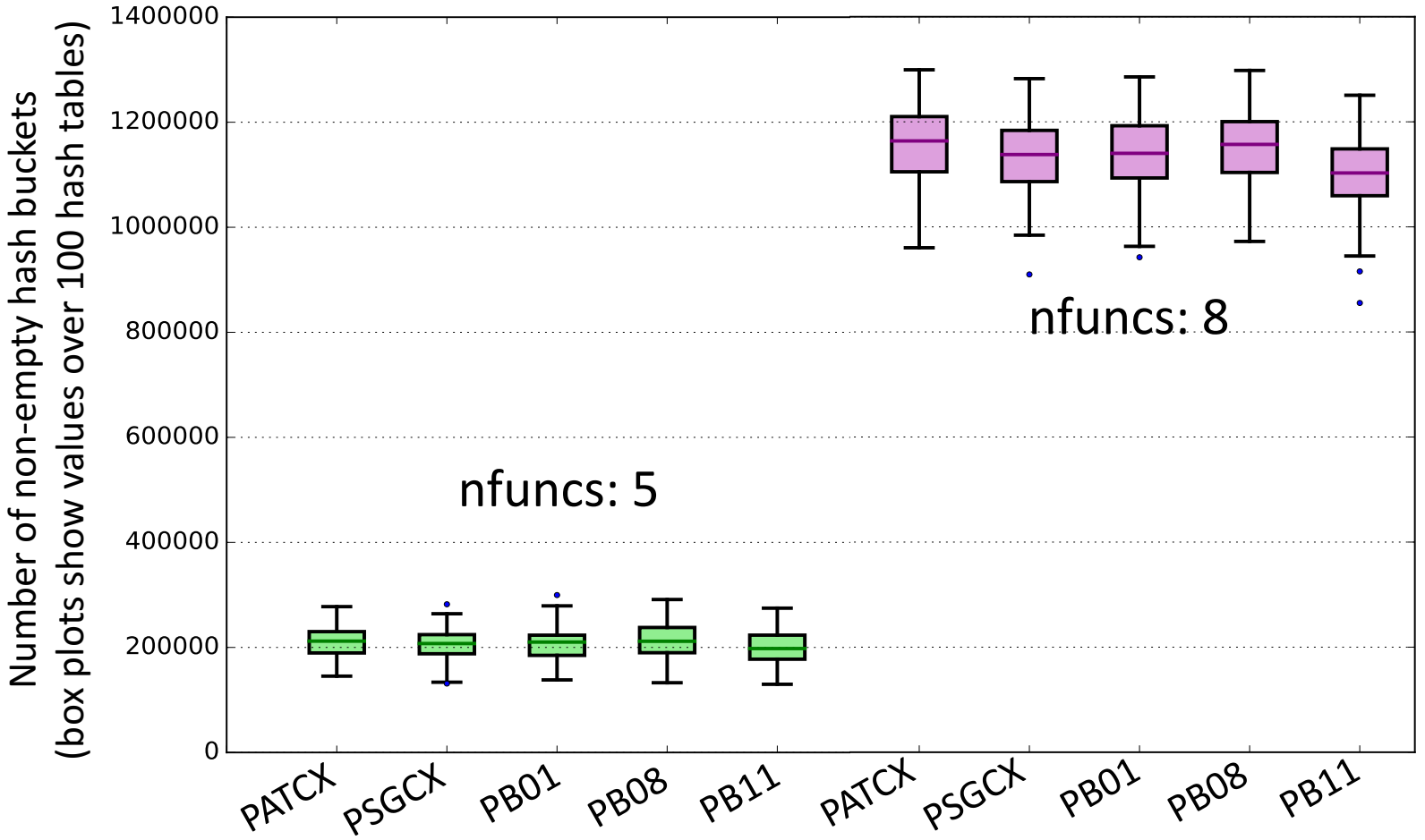
Runtime decreases with the increase of number of hash functions





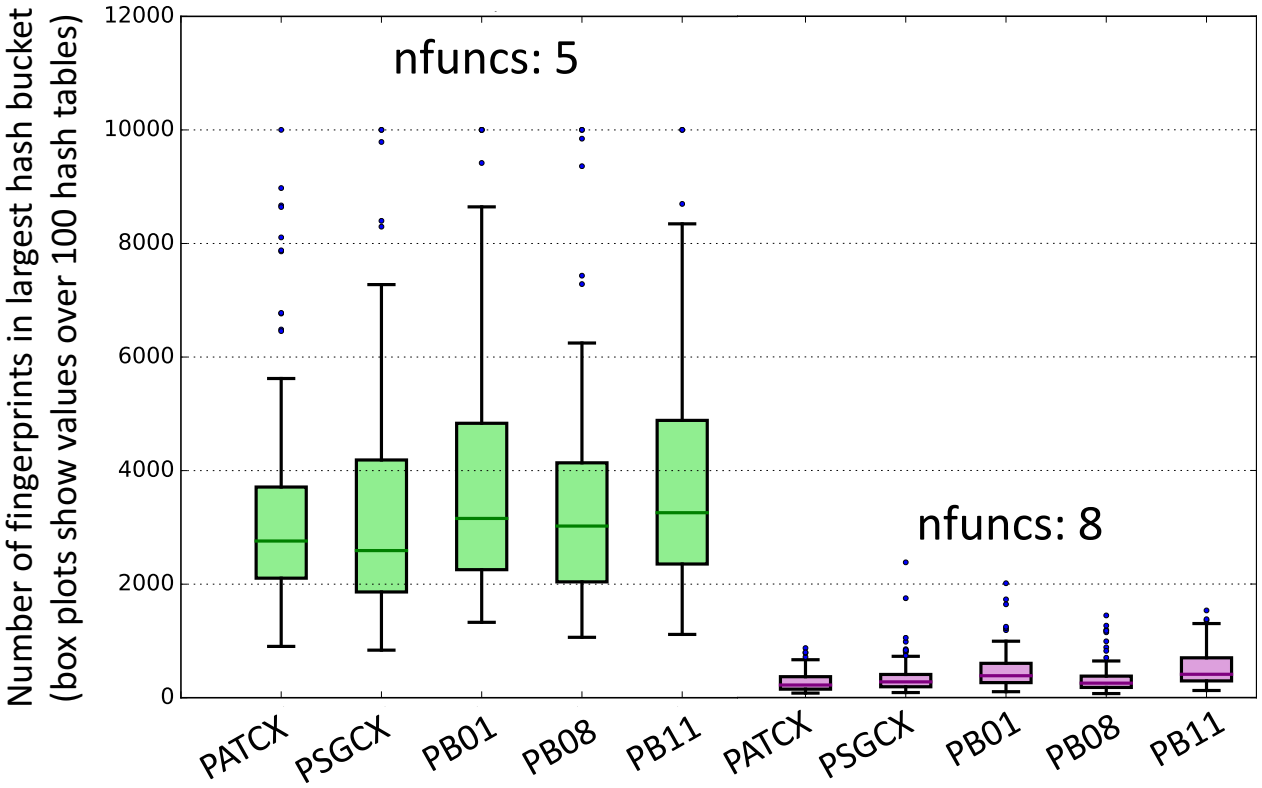
4.3 Similarity Search

Performance Impact of LSH parameters (explanation)



Larger nfuncs → more non-empty hash buckets…

Performance Impact of LSH parameters (explanation)



Larger "nfuncs" → more non-empty hash buckets, but "largest" hash buckets are smaller (i.e. fingerprints are more "spread out" within hash table) → faster runtime for similarity search but lower detection threshold (fingerprints must have higher Jaccard similarity for hash collisions).

LSH parameters and Detection Results

The **sensitivity** of the FAST detector depends on the combination of nfuncs and nvotes. The table below shows how the number of detections identified by multi-station FAST on the Iquique foreshock data set from Bergen & Beroza (2018a) changes as the nfuncs and nvotes parameters are varied. The first row (highlighted in blue) gives the parameter values used in Bergen & Beroza (2018a).

nfuncs	nvotes	# FAST detections	# template families (Kato et al, 2016) recovered by FAST (of 225)	# FAST detections in CSN catalog (of 571)
5	3	2788	198 (88%)	558 (98%)
6	3	1348	151 (67%)	480 (84%)
6	2	2233	186 (83%)	547 (96%)
7	3	883	118 (52%)	399 (70%)
7	2	1409	155 (69%)	480 (84%)
8	3	567	80 (36%)	323 (57%)
8	2	868	108 (48%)	395 (69%)
8	1	2084	185 (82%)	544 (95%)

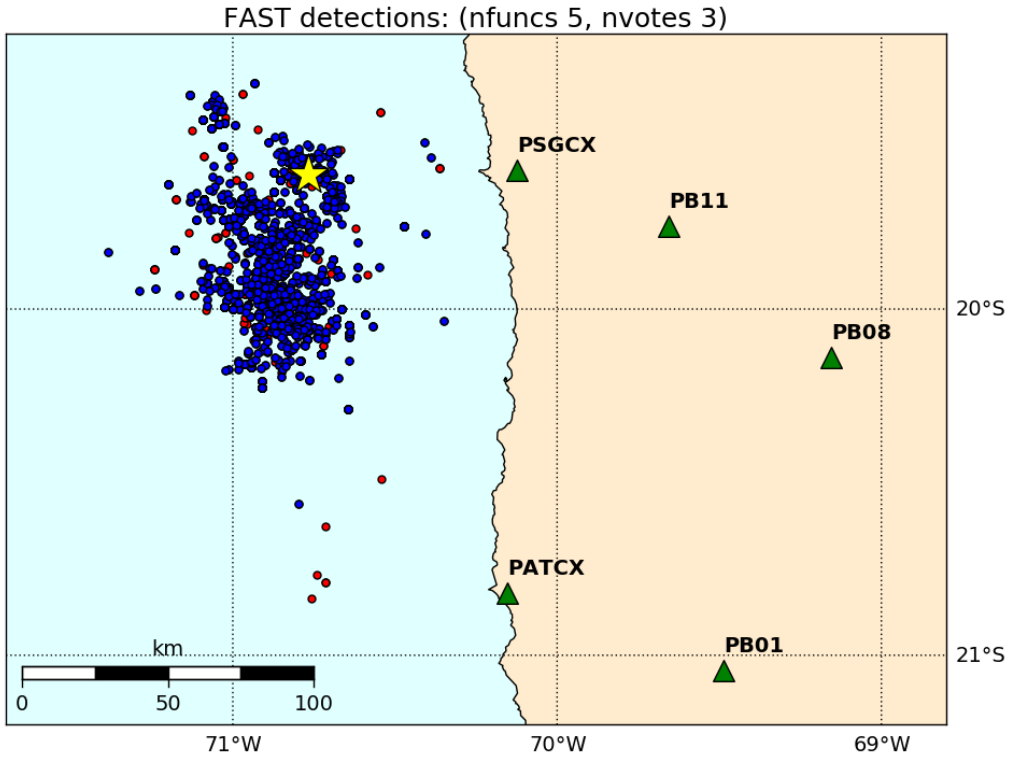
LSH parameters and Detection Results

The **size of the output** (# of fingerprint-pairs, i.e. # non-zeros in sparse similarity matrix) of the FAST detector depends on the combination of nfuncs and nvotes.

Increasing nvotes decreases the output size by increasing the detection threshold, while increasing nfuncs decreases the output size by lowering the detection threshold. Even though the number of detections for (nfuncs 5, nvotes 3) is slightly larger than for (nfuncs 8, nvotes 1), the latter has an output size that is an order of magnitude larger – this is due to a higher chance of spurious hash collisions associated with requiring only a single "vote" or collision out of 100 hash tables. Table shows output size for FAST detection results in Iquique case study from Bergen & Beroza (2018a).

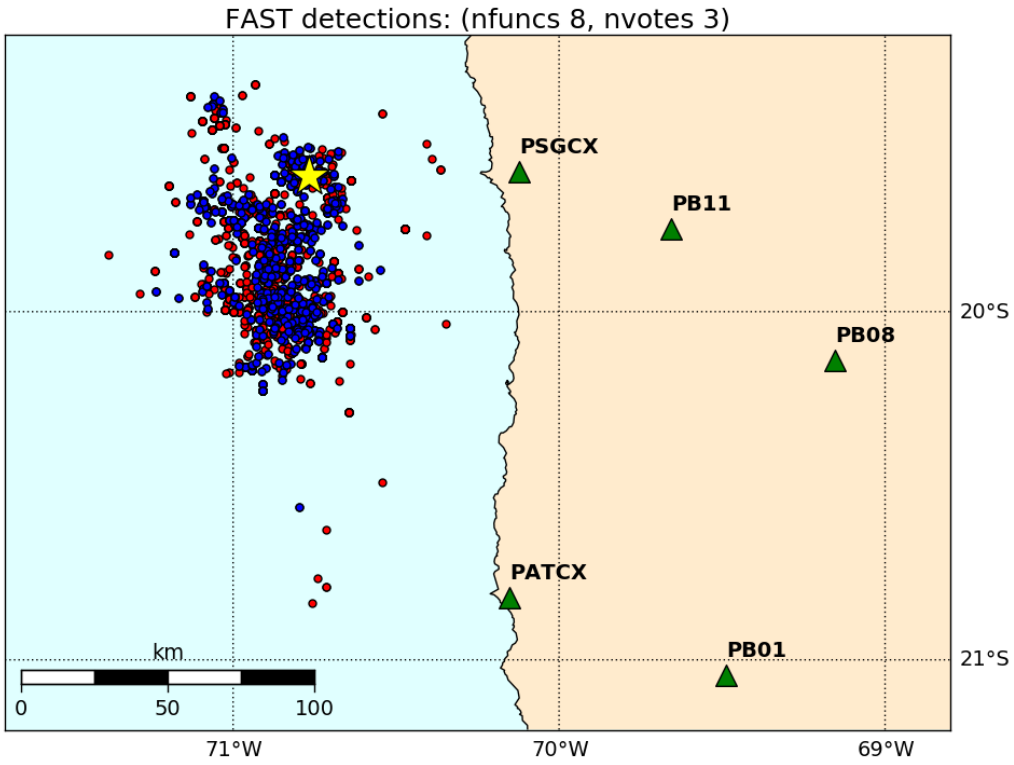
	# Fingerprint-pairs in single-station FAST output			**# FAST detections**
	PATCX.BHZ (minimum)	PB11.BHZ (maximum)	Average (over 5 stations)	
(nfuncs 8, nvotes 3)	0.22×10^6	3.04×10^6	1.45×10^6	567
(nfuncs 7, nvotes 3)	0.79×10^6	8.76×10^6	3.46×10^6	883
(nfuncs 6, nvotes 3)	0.31×10^7	2.09×10^7	1.06×10^7	1348
(nfuncs 5, nvotes 3)	**1.98×10^7**	**9.27×10^7**	**5.49×10^7**	**2788**
(nfuncs 8, nvotes 2)	1.28×10^6	9.97×10^6	5.27×10^6	868
(nfuncs 7, nvotes 2)	0.47×10^6	2.98×10^7	1.49×10^7	1409
(nfuncs 6, nvotes 2)	2.14×10^7	8.42×10^7	4.84×10^7	2233
(nfuncs 8, nvotes 1)	0.68×10^8	1.77×10^8	1.20×10^8	2084

LSH parameters and Detection Results



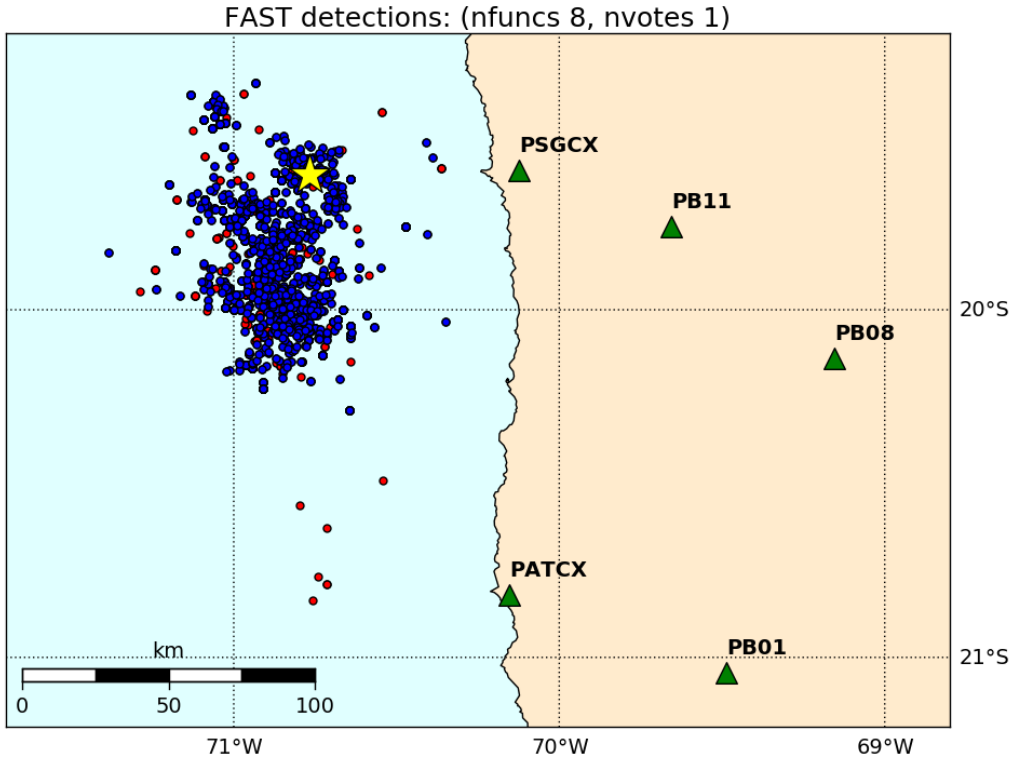
Locations of FAST detections in Kato et al (2016) template matching catalog or in local seismicity catalog (CSN) for detection with LSH parameter values: **nfuncs: 5, nvotes: 3**

LSH parameters and Detection Results



Locations of FAST detections in Kato et al (2016) template matching catalog or in local seismicity catalog (CSN) for detection with LSH parameter values: **nfuncs: 8, nvotes: 3**

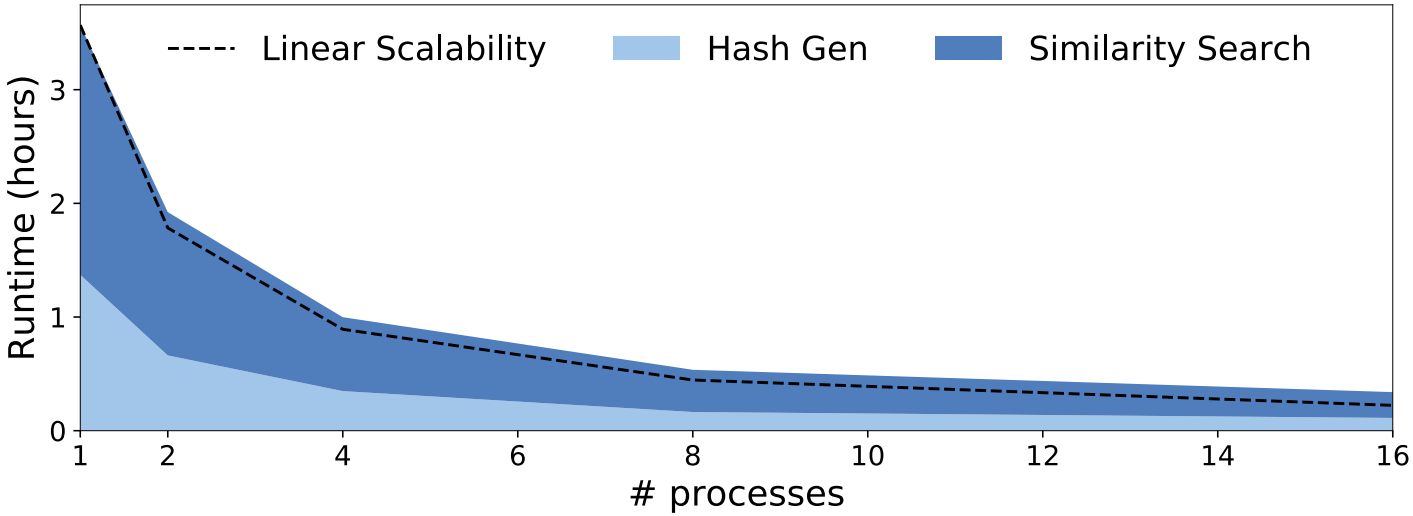
LSH parameters and Detection Results



Locations of FAST detections in Kato et al (2016) template matching catalog or in local seismicity catalog (CSN) for detection with LSH parameter values: **nfuncs: 8, nvotes: 1**

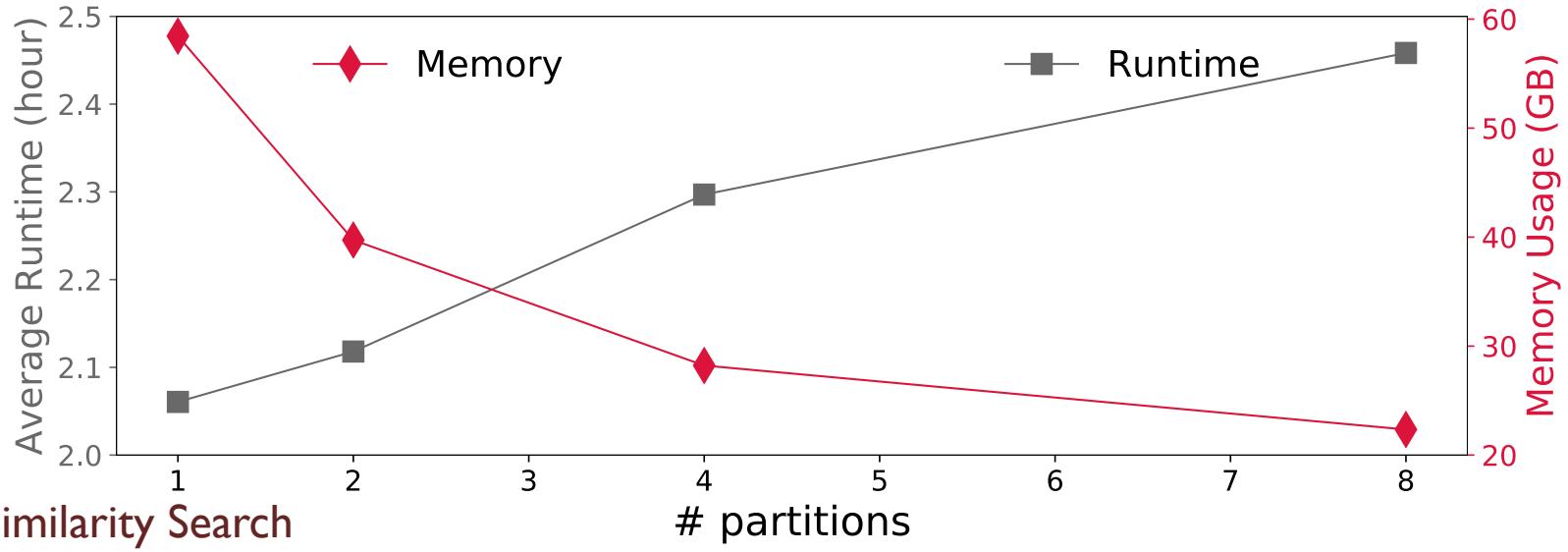
LSH Parameter Guidance (Performance related)

- ncores: Number of processes for parallel processing
  - For large continuous data sets (>months): use as many as your machine allows
  - Runtime decreases inverse proportionally with the increase of number of threads



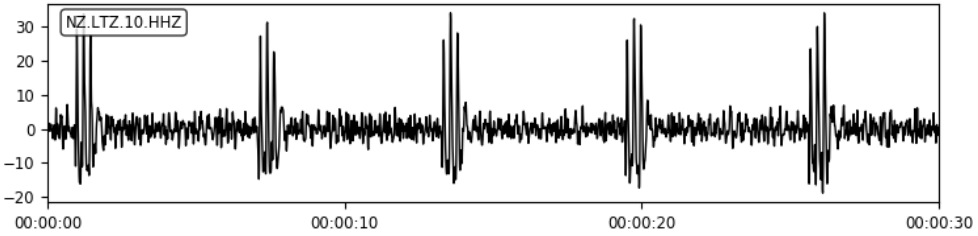
LSH Parameter Guidance (Performance related)

- num_partitions: Number of partitions for LSH database
  - LSH database can be enormous, especially for large continuous data sets
  - FAST is intended to run on Linux clusters with lots of memory (> 64 GB); figure out how much memory is on your machine
    - If data set is small (days-weeks), use num_partitions= 1 (entire database fit in memory)
    - Use num_partitions > 1 for larger data sets, so that each partition fits in memory



LSH Parameter Guidance (Performance related)

- noise_freq: (Occurrence filter) Frequency threshold above which fingerprints will be filtered out as correlated noise
  - Consider turning on this filter if some of the stations/channels produce orders of magnitude more outputs or take significant longer than its counterparts
  - The slow down might be caused by the large number of matches generated from persistent background noise (see figure below as an example)



Example of correlated noise

LSH Parameter Guidance (Performance related)

- Increase num_partitions to capture correlated noise that only appear in short segments of the input data
- Example parameters
  - Input: 1 year of time series data
    - noise_freq=0.01 num_partitions=12
    - Filter out fingerprints that matches over 1% of other fingerprints in the current partition (partition length = 1 year / 12 = 1 month)
  - Input: 1 month of time series data
    - noise_freq=0.005 num_partitions=30
    - Filter out fingerprints that matches over 0.5% of other fingerprints in the current partition (partition length = 1 month / 30 = 1 day)
- Performance benefits
  - Can reduce the similarity search output size by orders of magnitude as well as improve runtime
  - Reduced output size further improves the postprocessing runtime

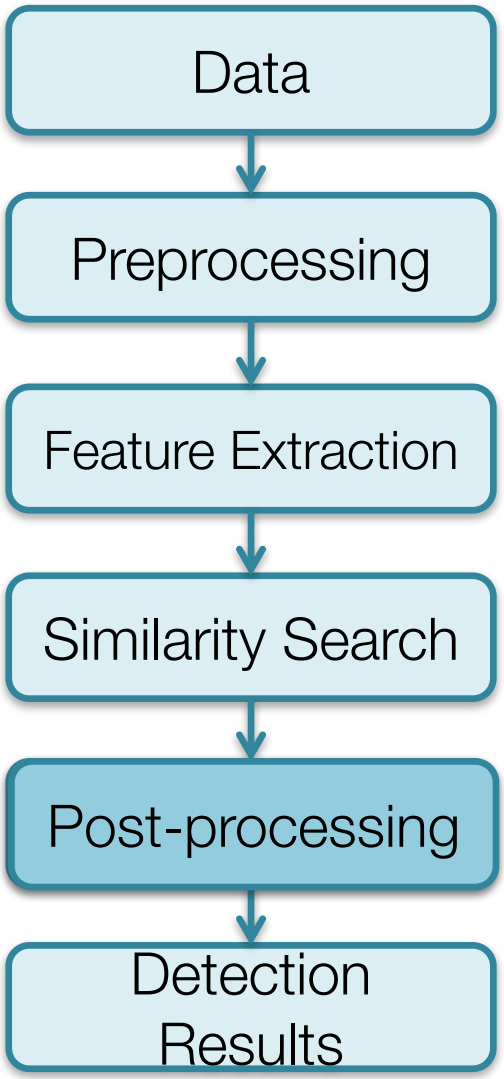
FAST Similarity Search Output Format

- Naming convention for text output files
  - One file, regardless of npart:
    `candidate_pairs_${STATION}_${CHANNEL}_merged.txt`
  - Example text file (output):
    `candidate_pairs_CDY_EHZ_merged.txt`
- Content of output file:
  - list of pairs of similar fingerprints and their similarity, sorted in increasing dt order: dt = index1-index2, index1, sim
  - index1 is index of first fingerprint (according to global index)
  - index2 is index of second fingerprint (according to global index)
  - sim is "FAST similarity": Number of hash tables (out of b=ntbl) containing fingerprints index1 and index2 in same bucket; should be ≥ (v=nvote). Note: not normalized as the fraction divided by b.
- To get fingerprint times
  - get global start time t0 from global_idx_stats.txt, fingerprint lag dt_fp in seconds
  - time1 = t0 + dt_fp*index1, time2 = t0 + dt_fp*index2

Outline

1. FAST Overview
2. Requirements and Installation
3. Hector Mine Tutorial
4. How to set parameters
    1. Input and Preprocessing
    2. Fingerprint
    3. Similarity Search
    4. Network Detection
    5. Output
5. References
6. Supplementary

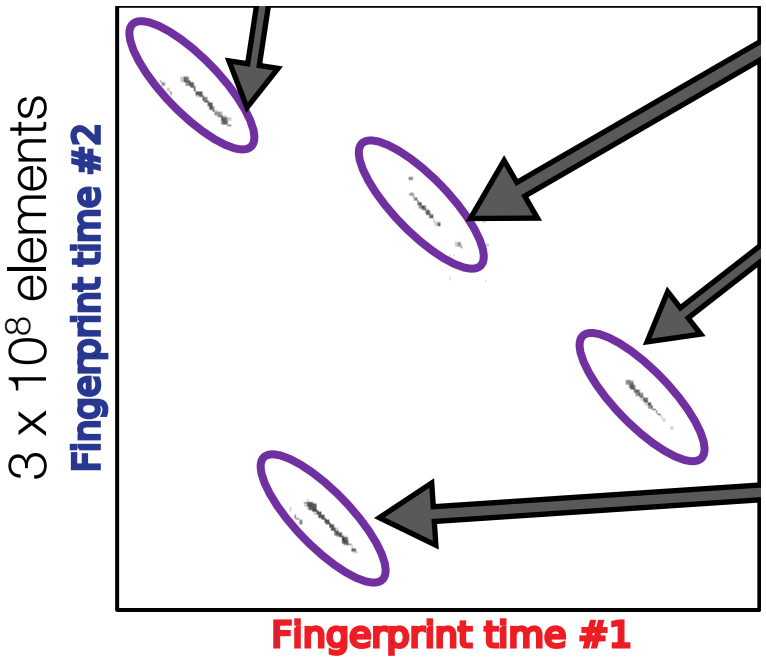
Postprocessing Overview



FAST output: sparse similarity matrix

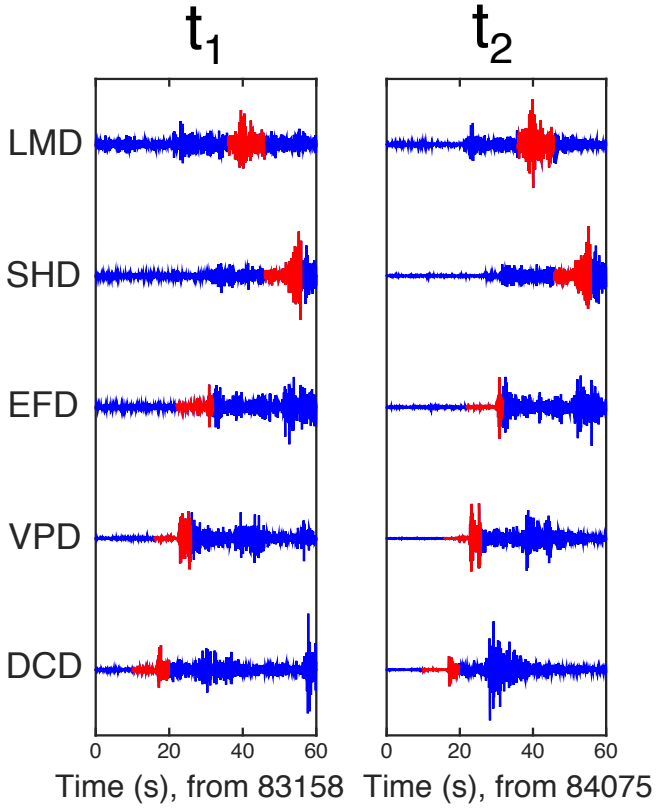
9×10^{16} matrix

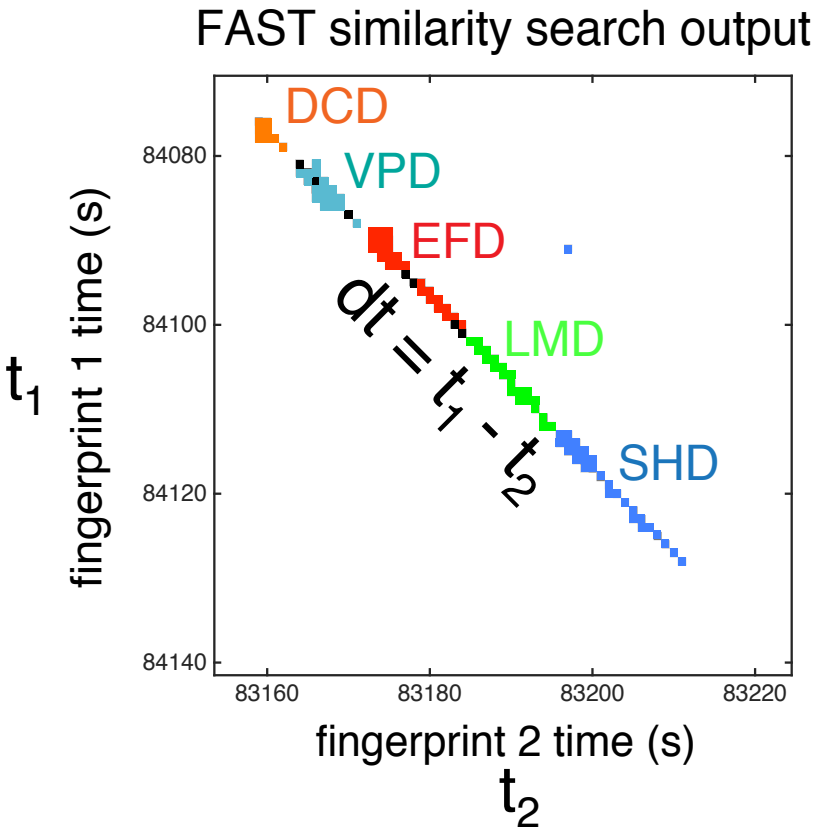
(<0.0001% elements filled)



Association: Detect earthquakes over a seismic network

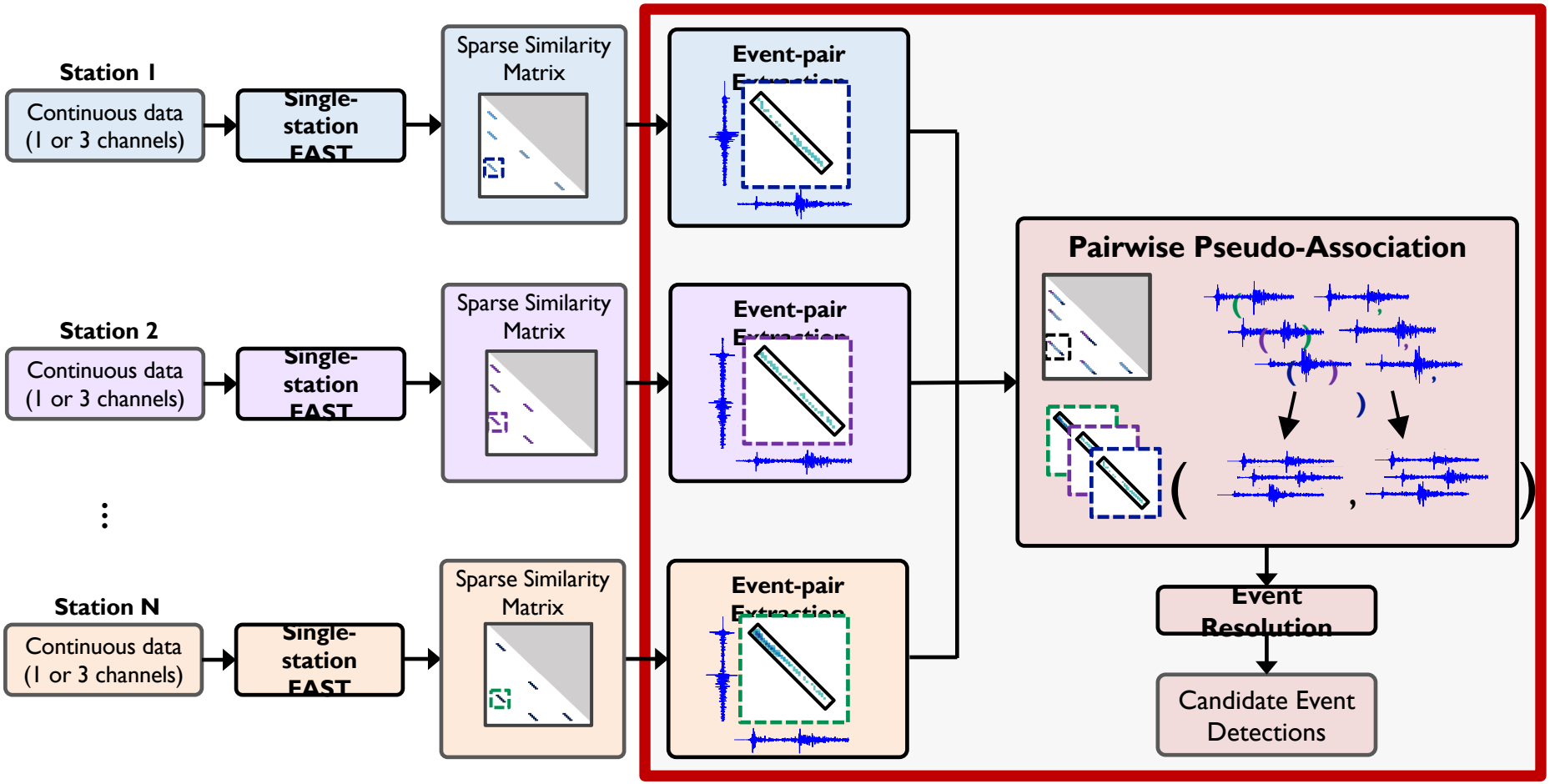
Earthquake pair at different stations: consistent inter-event time dt
Reduce false detections





Bergen and Beroza (2018)

Network (Multi-station) Detection with FAST

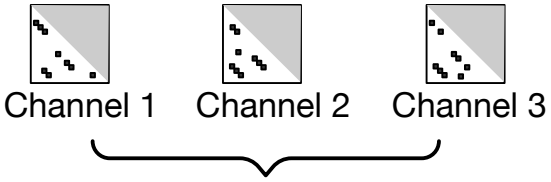


4.4 Network Detection

Bergen and Beroza (2018)

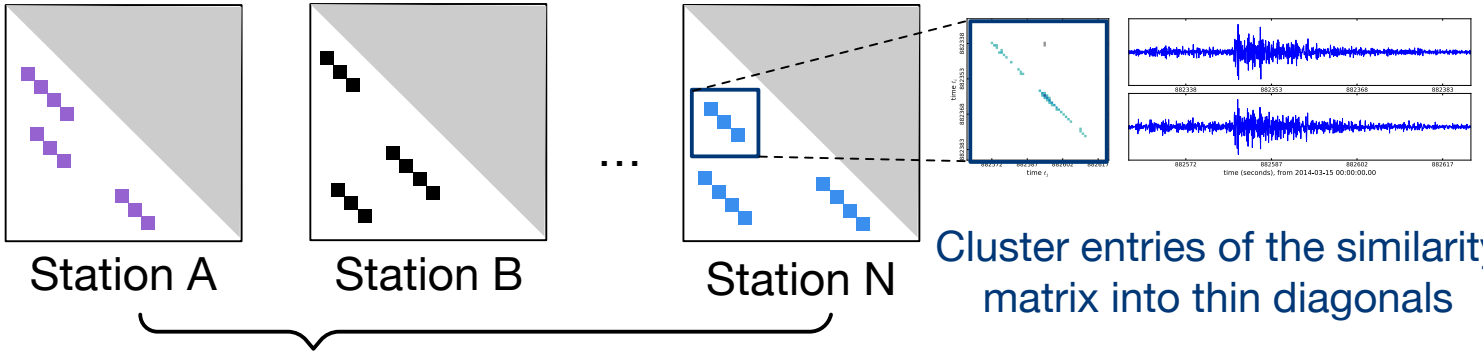
Network (Multi-station) Illustration

Channel Level



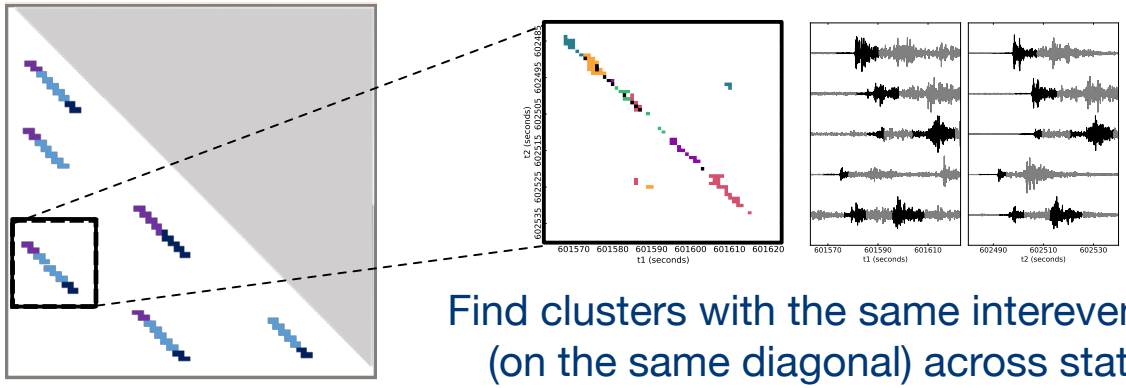
Combine similarity matrix from channels of the same station

Station Level



Cluster entries of the similarity matrix into thin diagonals

Network Level



Find clusters with the same interevent time (on the same diagonal) across stations

Event-pair extraction parameters

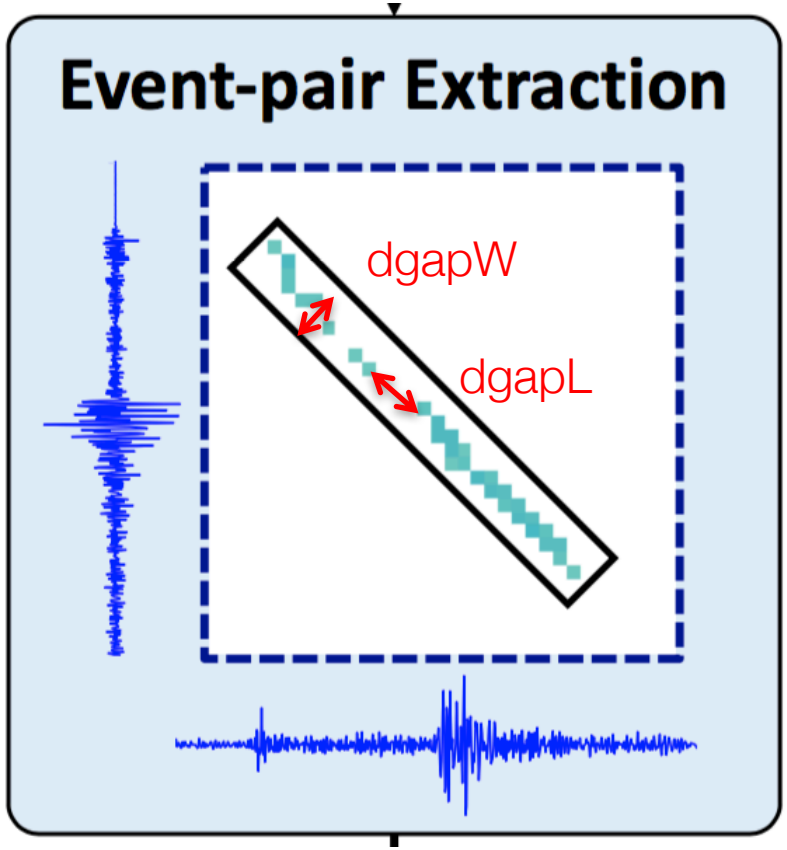
```
”network": {
        "dgapL": 3,
        ”dgapW": 3,
        ”num_pass": 2,
        ”ivals_thresh": 6,
        “min_dets": 4,
        ”min_sum_multiplier": 1,
        "max_width": 8,
},
```

Longest allowed along-diagonal time gap (samples)
Longest allowed cross-diagonal time gap (samples)
Adjacent diagonal merge iterations
Minimum similarity (number of votes)
Minimum number of fingerprint-pairs in a cluster
Minimum total similarity multiplier for a cluster
Maximum bounding box width (samples)

Guidelines for setting parameters:

- ivals_thresh = nvote**, unless initial threshold is too low. Better approach is increasing** min_sum_multiplier **(i.e.** min_sum_multiplier = updated_nvote / nvote **) which effectively increases fingerprint-pair threshold to** updated_nvote **with better thresholding that takes advantage of structure of event-pairs**
- min_dets = 4-6
- dgapL**: time interval equivalent to largest expected P-S gap (e.g. 10-20 seconds)**
- dgapW: **should be small, equivalent to a few seconds (3-4 seconds)**

```
”network": {
        “max_fp": 74797,     Max fingerprint index over all stations
        ”dt_fp": 1.0,        Fingerprint lag (s)
        "dgapL": 3,          Longest allowed along-diagonal time gap (samples)
        ”dgapW": 3,          Longest allowed cross-diagonal time gap (samples)
        ”num_pass": 2,       Adjacent diagonal merge iterations
    },
```



Event-Pair Extraction Parameters (1 station)

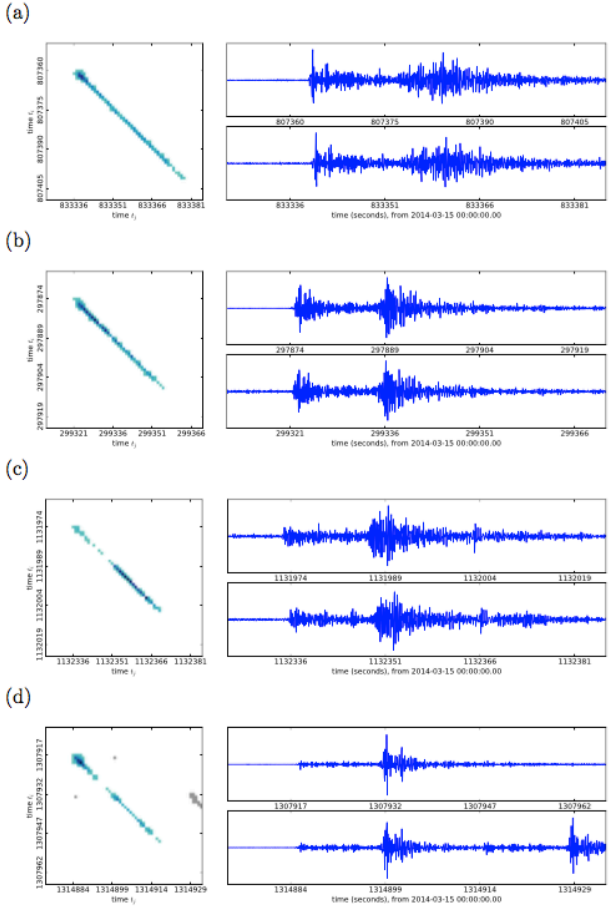
- dgapL, dgapW: determine whether to keep as 1 cluster or split into 2 clusters
  - Multiply dgapL, dgapW by dt_fp to get values in seconds
- Num_pass=2 is good default (3 is also ok, but takes longer)

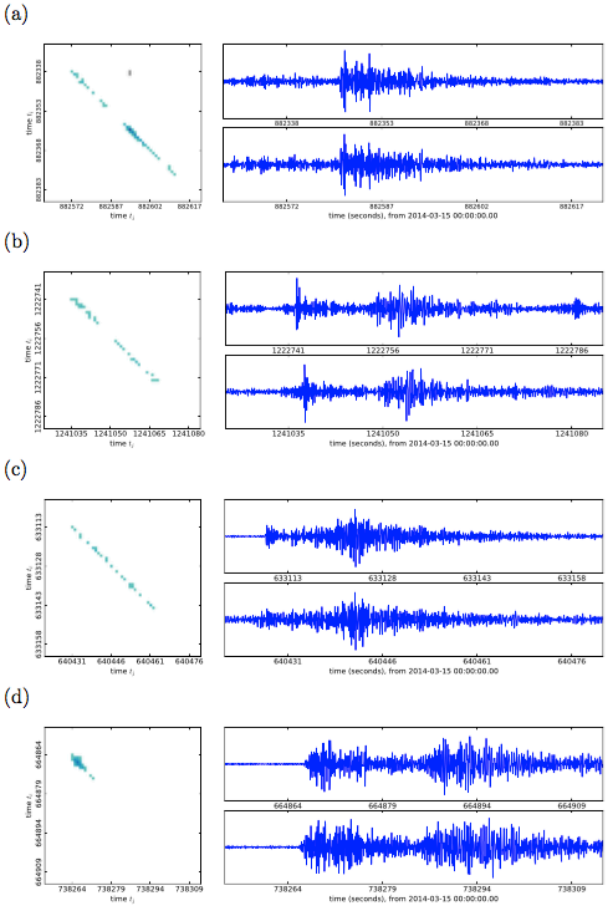
Event Pair Extraction Examples

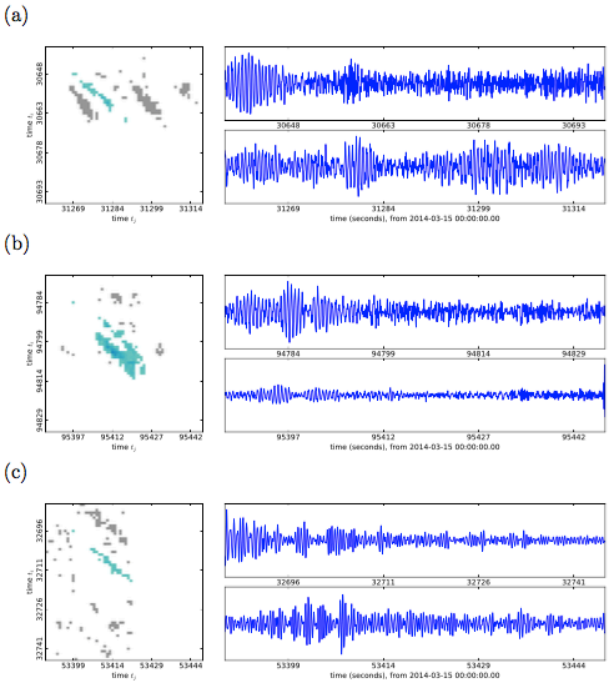
Highly similar events

Not so similar, noisier events

Similar Noise







Bergen and Beroza (2018)

```
”network": {
”ivals_thresh": 6,
“min_dets": 4,
”min_sum_multiplier": 1,
"max_width": 8,
},
```

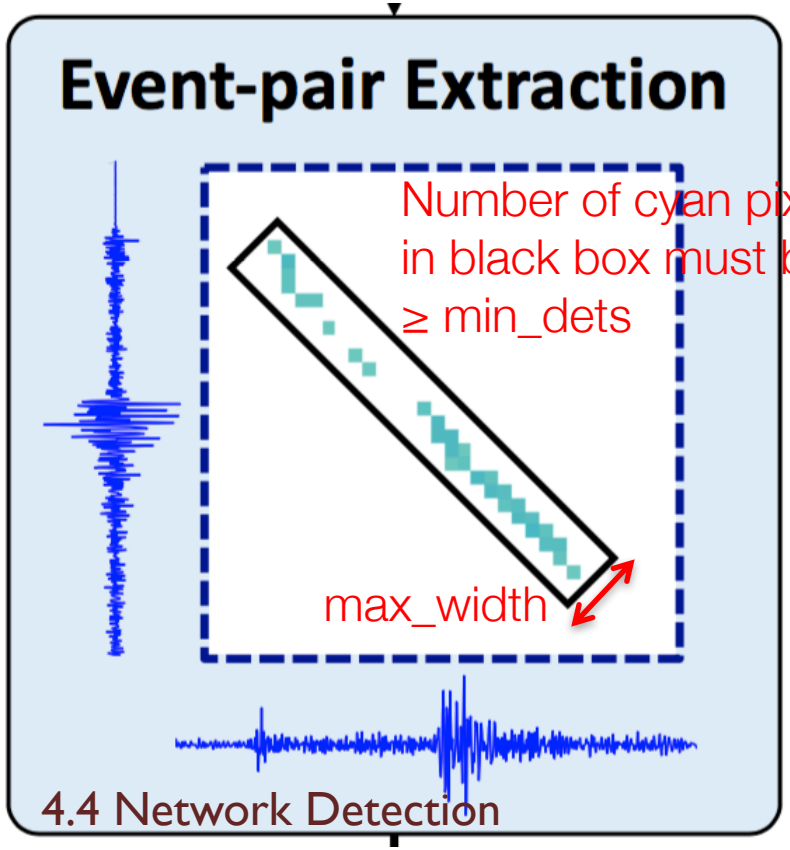
Minimum similarity (number of votes)

Minimum number of fingerprint-pairs in a cluster

Minimum total similarity multiplier for a cluster

Maximum bounding box width (samples)

Similarity of each cyan pixel in black box must exceed ivals_thresh



Event-Pair Pruning Parameters (1 station)

- Set higher thresholds on similarity in order to identify an event-pair cluster
- Minimum total similarity threshold: ivals_thresh * min_dets * min_sum_multiplier
- max_width: 8 is a good default value, probably don’t need to change
  - multiply by dt_fp to get value in seconds

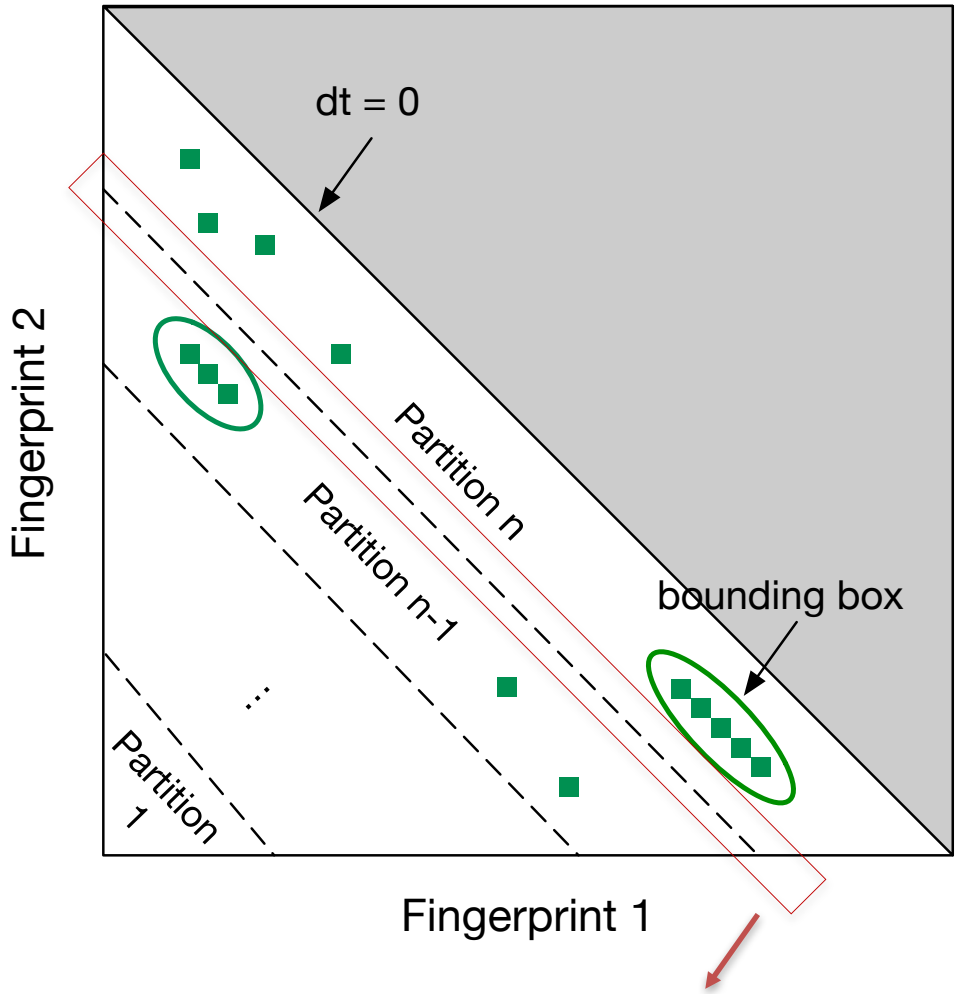
Network Detection Performance Parameter Guidance

```
"performance": {
        "partition_size": 2147483648,
        "num_cores": 4
    },
```

- partition_size: Maximum size of each partition (bytes), if entire list of similarity search output pairs does not fit into memory
- num_cores: Number of threads for parallel processing (event-pair extraction only)

Network Detection Performance Parameter Guidance

- The similarity matrix is partitioned along the diagonals (dt)
- Each core can perform event-extraction in parallel
- Set partition_size to smaller than total size of the output divide by num_cores to make sure that each core has at least one partition to process

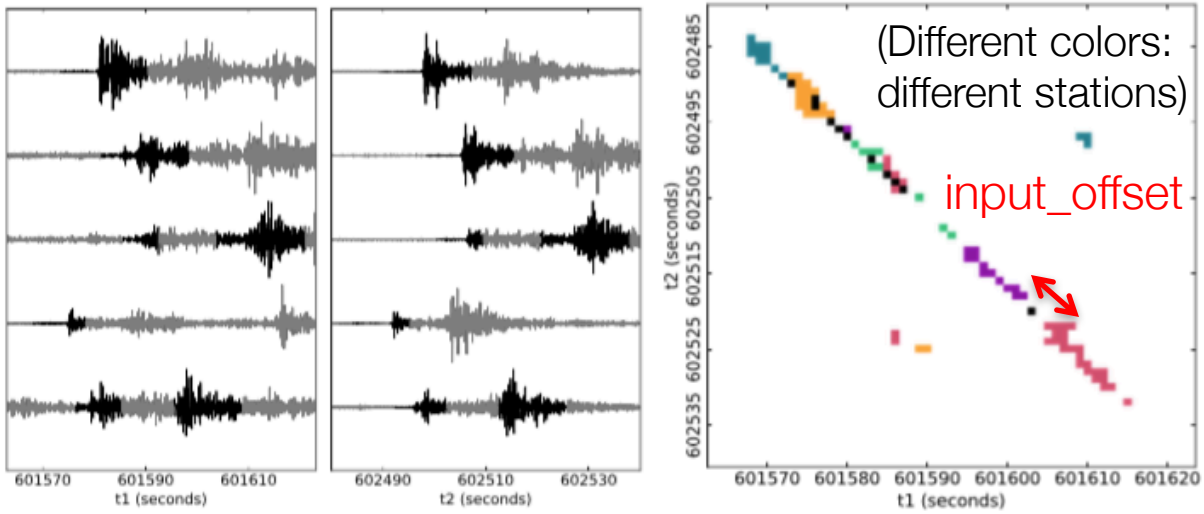


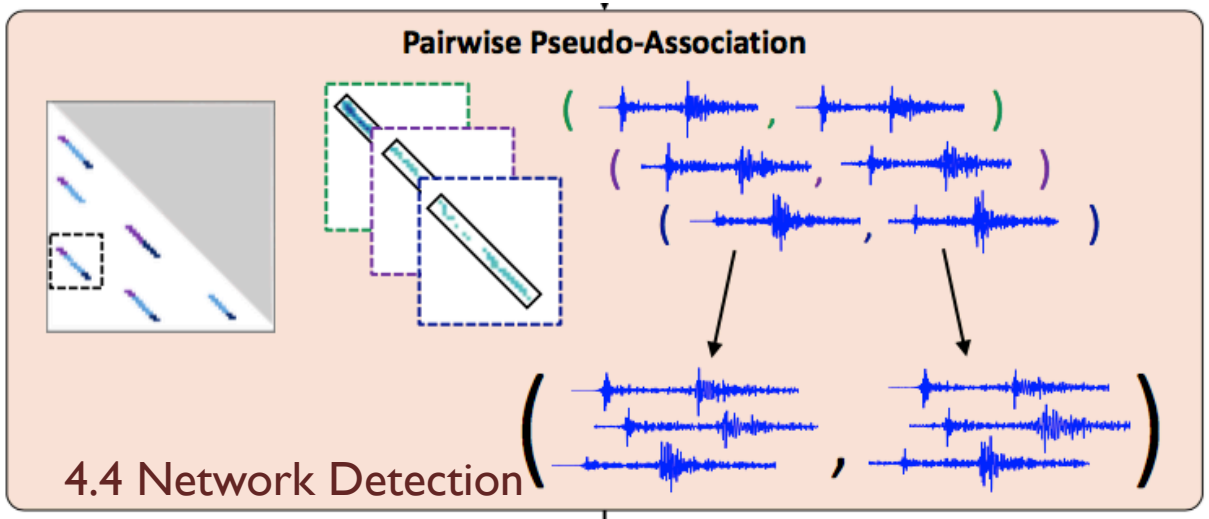
Pseudo-Association Parameters (Multiple stations)

```
"network": {
    "nsta_thresh": 2,
    "input_offset": 3,
},
```

Minimum number of stations for detection

Arrival time constraint: maximum time gap (samples)





- Set nsta_thresh low to begin, can increase threshold later
- nsta_thresh=1: single station detection not yet thoroughly tested
- Set input_offset to be largest expected time gap between S wave at 1 station and P wave at another station
  - Multiply by dt_fp to get value in seconds

Pseudo-Association Parameters (Multiple stations)

```
”network": {
      ”nsta_thresh": 2,
      “input_offset": 3,
},
```

Minimum number of stations for detection

Arrival time constraint: maximum time gap (samples)

- input_offset: arrival time constraint: maximum time gap (samples)
  - input_offset is the longest time period that can elapse with no active detections (for a given – event-pair): after a gap of more than input_offset samples within a network event-pair, pseudo-association will automatically create a new network event-pair.
  - A reasonable upper-bound on input_offset is the largest expected S-P time for any station in the network, or the largest difference in P arrival times between any pair of stations in the network, whichever is larger. Usually, input_offset can be shorter than this upper-bound, but the value should not be smaller than the time gap parameter used in event-pair extraction: input_offset > dgapL.
  - Note that because all event-pairs are only grouped together into network event-pairs only if they have nearly identical inter-event times, there is limited risk of falsely associating event-pairs or of being unable to resolve overlapping events (with detections separated by a time interval of less than input_offset) due to selecting the value of input_offset that is somewhat longer than necessary. However, input_offset should not be set to a value that is unnecessarily large (e.g. 60+ seconds) because this does increase the chance of spurious associations of unrelated event-pairs observed at different stations in the network.
  - **For most data sets, the value of network time gap parameter should lie within the range, input_offset = 15-40 seconds**

Event resolution implementation

- Event resolution is applied separately to each station using detections from each station that meet network detection threshold (e.g. observed at minimum number of station)
- In this implementation, event resolution method is relatively simple to minimize memory usage for large data sets. Specifically event resolution does not keep information about structure of pairwise detections, only whether there was a detection (in any event-pair) for each time stamp.
- Thus in this version of the code, events that are overlapping in time at a single station with be resolved to a single event. In order to tease out two separate events that overlap in time at a single station, the user will need to go back to the network detection output from pairwise pseudo-association. Two events overlapping in time at a single station will belong to different network event-pairs (if there are two detections at the same station in the same network event-pair, then these are not overlapping events, but different phases of the same event).
- See next slide for event resolution algorithm.

Event resolution implementation algorithm:

1. Events are separately resolved for each station to create a master event list for each station. For each station:
   a) For every time index, obtain a count of the number of network event- pairs (from pairwise pseudo-association) that include an event that occurs at this station at this time (and keep track of associated network event-pair IDs).
   b) Identify events by looking for time gaps between detection counts. Events are defined as contiguous time intervals during which detections were observed at this station and labeled according to the initial time of the interval. This will be the master event list for the station. (Note: additional processing may be necessary if number of false positives or density of events is too high to result in reasonable gaps between events.)
   c) For each event, identify the list of the network event-pair IDs associated with this event.

2. For each network event-pair from pairwise pseudo-association output:
   a) Map partial event detections in event-pair to corresponding master event for that station.
   b) Output two network event entries for each pair, one for each of the two events that belong to the pair. Output for each of the two events should include time associated with master event for each station, and any summary statistics associated with the network event. If event is not observed at every station, list time as nan for missing stations.

3. Remove duplicate detections from list of network event detections, aggregate any summary statistics for duplicate events, and keep a tally of number of times event was duplicated (corresponds to number of similar events

Sample Parameter Settings: *2014 M8.2 Iquique foreshock sequence* (from Bergen & Beroza, 2018)

Parameters	Value	Explanation
dgapL (g_L)	15 s	15-40s is an appropriate range for data (40s is more conservative) with events 50-200km from stations; parameter reflects largest allowable gap in detections within an event, gaps often corresponds to period between P and S arrivals - should be adjusted based on expected S-P time.
dgapW (g_W)	3 s	2-5s is reasonable range for most data sets
num_pass (p)	3	1-3 passes ok – 3 leads to better groups for irregular clusters (which are usually false detections)
ivals_thresh	3	"ivals_thresh" can be used to effectively raise "nvote" after running similarity search, but in this data set we keep ivals_thresh = nvote,
min_dets	5	Eliminates very small event-pair clusters
min_sum_multiplier	2	Minimum total similarity, $v_{min}^{(C)} = 30$; eliminates event-pair clusters of min_det fingerprint-pairs each with similarity ivals_thresh – to be accepted event-pair cluster either needs higher total similarity (equivalent to 5 fingerprint-pairs of similarity 6 or 10 fingerprint-pairs of similarity 3).
max_width	8 s	In practice very wide clusters (> 5-6 s width) tend to be due to false positives from correlated noise
input_offset (g_N)	20 s	15-60s is an appropriate range
nsta_thresh	4	4 of 5 stations for high confidence in detections → resulting false discovery rate is <1%

Network Detection Outputs

- Event resolution: final step after pseudo-association
  - Pairs of similar fingerprints -> list of event detections
- Network Detection Output (text file with labeled columns)
  - Example (ranked in descending order of 'peaksum'): `7sta_2stathresh_detlist_rank_by_peaksum.txt`
  - First (num_sta=number of stations) columns: starting fingerprint index at each station (time information)
    - Outputs "nan" if not observed at a particular station
  - dL: Maximum length (samples) along diagonal, over all event-pairs containing this event
  - nevents: Number of other events 'linked' to (similar to) this event
  - nsta: Number of stations over which other events are similar to this event
  - tot_ndets: Total number of fingerprint-pairs (pixels) containing this event, over all event-pair clusters, over all stations
  - max_ndets: Maximum number of fingerprint-pairs (pixels) containing this event, over all event-pair clusters, over all stations
  - tot_vol: Total sum (or 'volume') of all similarity values (added over all stations), over all event-pairs containing this event
  - max_vol: Maximum sum (or 'volume') of all similarity values (added over all stations), over all event-pairs containing this event
  - max_peaksum: Maximum similarity value (added over all stations), over all event-pairs containing this event

Outline

1. FAST Overview
2. Requirements and Installation
3. Hector Mine Tutorial
4. How to set parameters
    1. Input and Preprocessing
    2. Fingerprint
    3. Similarity Search
    4. Network Detection
    5. Output
5. References
6. Supplementary

References

(FAST Overview) C. Yoon, O. O'Reilly, K. Bergen and G. C. Beroza (2015) Earthquake detection through computationally efficient similarity search, *Sci. Adv. 1, e1501057.*

(Fingerprint) K. Bergen, C. Yoon and G. C. Beroza (2016) Scalable Similarity Search in Seismology: A New Approach to Large-Scale Earthquake Detection, *9th International Conference on Similarity Search and Applications, Tokyo, Japan, 1-8.*

(Network Detection) K. Bergen and G.C. Beroza (2018) Detecting Earthquakes over a Seismic Network using Single-Station Similarity Measures, *Geophys. J. Int., doi:10.1093/gji/ggy100*

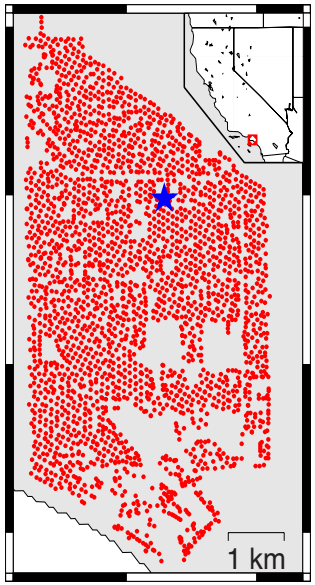
(Application of FAST) C. Yoon, Y. Huang, W. L. Ellsworth and G. C. Beroza (2017) Seismicity During the Initial Stages of the Guy-Greenbrier, Arkansas, Earthquake Sequence, *J. Geophys. Res., doi:10.1002/2017JB014946*

(Implementation and Performance) K. Rong, C. Yoon, K. Bergen, H. Elezabi, P. Bailis, P. Levis and G. C. Beroza (2018) Locality-Sensitive Hashing for Earthquake Detection: A Case Study Scaling Data-Driven Science, *arXiv:1803.09835.*

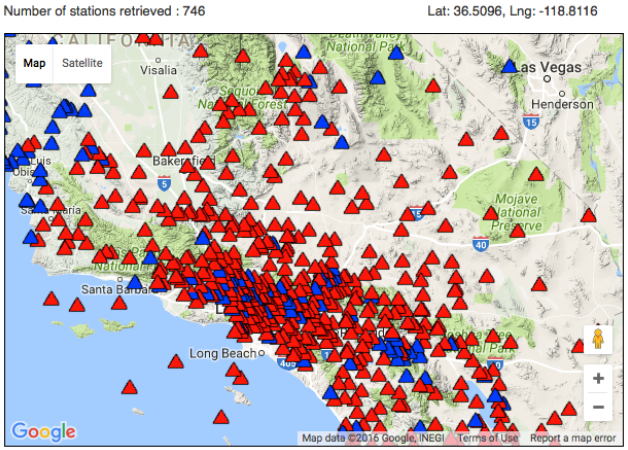
Outline

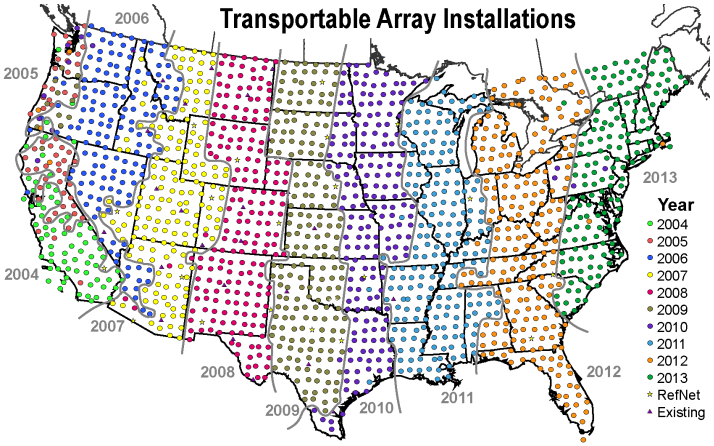
1. FAST Overview
2. Requirements and Installation
3. Hector Mine Tutorial
4. How to set parameters
    1. Input and Preprocessing
    2. Fingerprint
    3. Similarity Search
    4. Network Detection
    5. Output
5. References
6. Supplementary

Seismology has big data sets

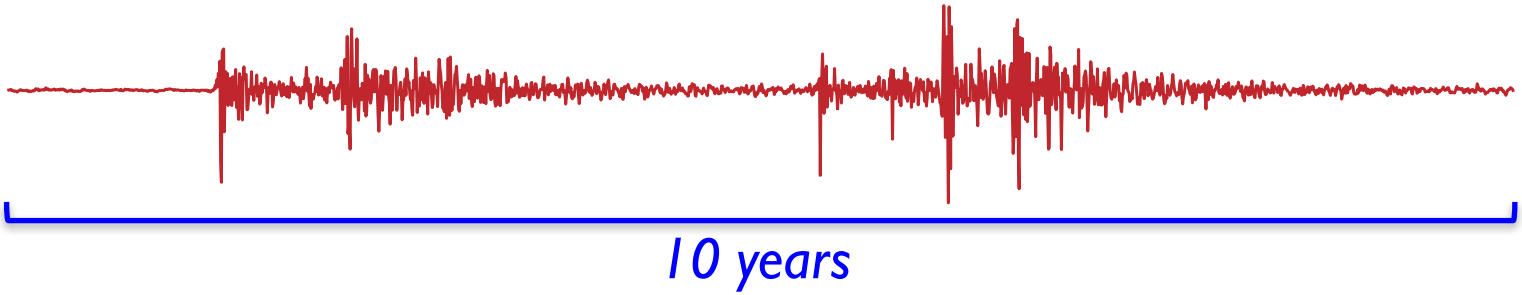


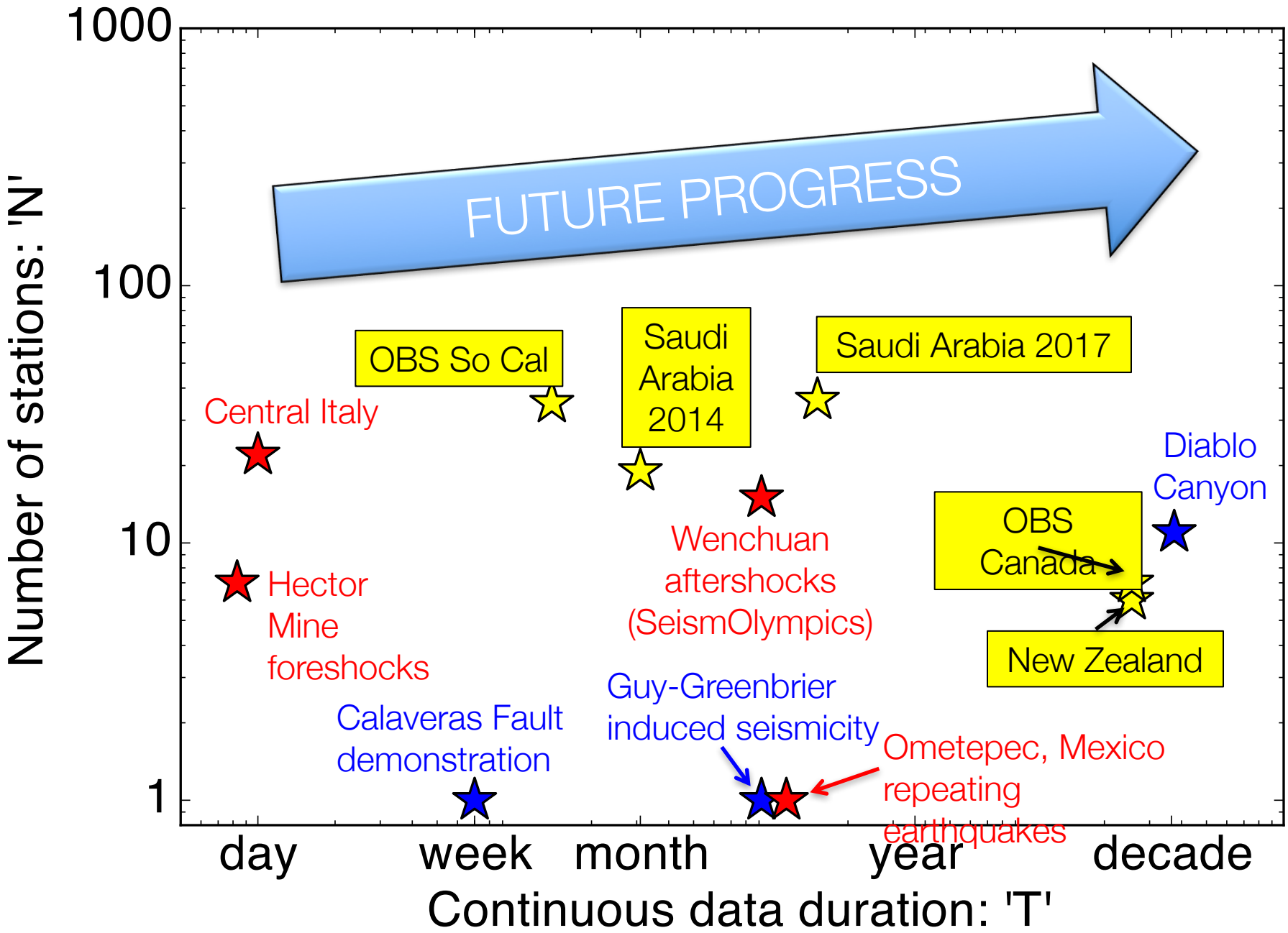
Large-N: big seismic networks





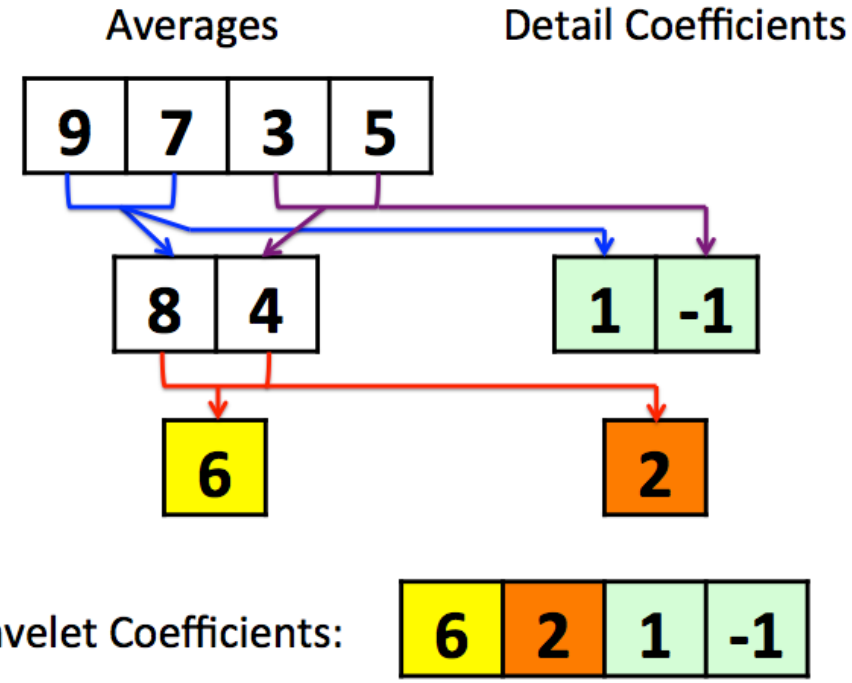
Large-T: continuous seismic data over long times




1000
100
10
1
Number of stations: 'N'
FUTURE PROGRESS
OBS So Cal
Saudi Arabia 2014
Saudi Arabia 2017
Central Italy
Diablo Canyon
OBS Canada
Wenchuan aftershocks (SeismOlympics)
Hector Mine foreshocks
New Zealand
Guy-Greenbrier induced seismicity
Calaveras Fault demonstration
Ometepec, Mexico repeating earthquakes
day
week
month
year
decade
Continuous data duration: 'T'

Example: 1D Haar Discrete Wavelet Transform

Decomposition



Recursive algorithm ➔ fast

Synthesis

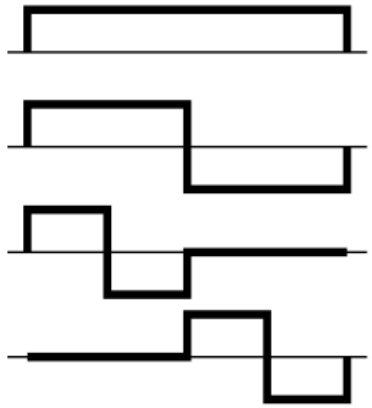
$$\mathcal{I}(x) = c_0^0\,\phi_0^0(x) + d_0^0\,\psi_0^0(x) + d_0^1\,\psi_0^1(x) + d_1^1\,\psi_1^1(x)$$

$$= 6 \times$$

$$+ 2 \times$$

$$+ 1 \times$$

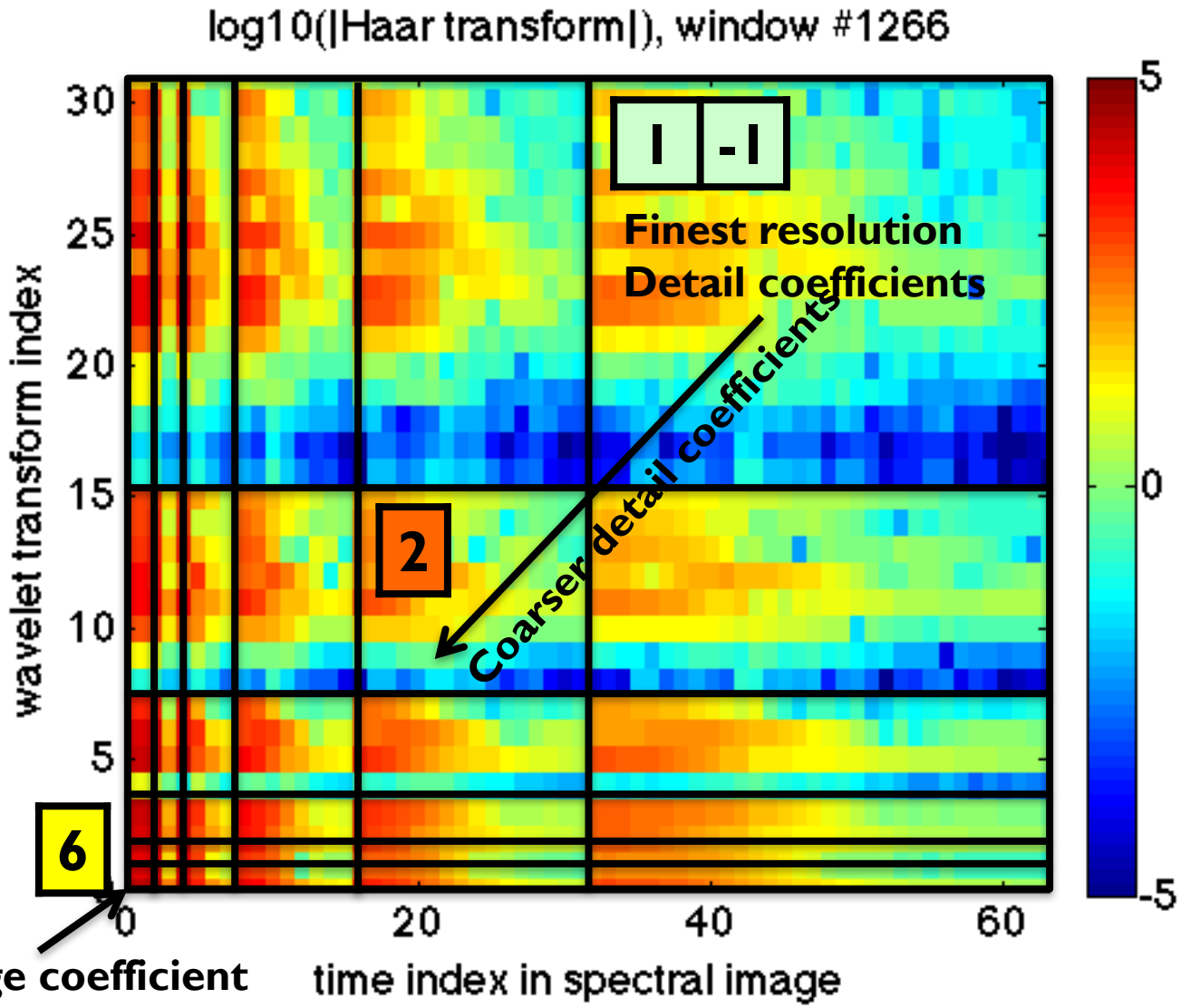
$$+ -1 \times$$



Haar basis: Square waves with different resolutions

Stollnitz et al. (1995)

2D Discrete Wavelet Transform

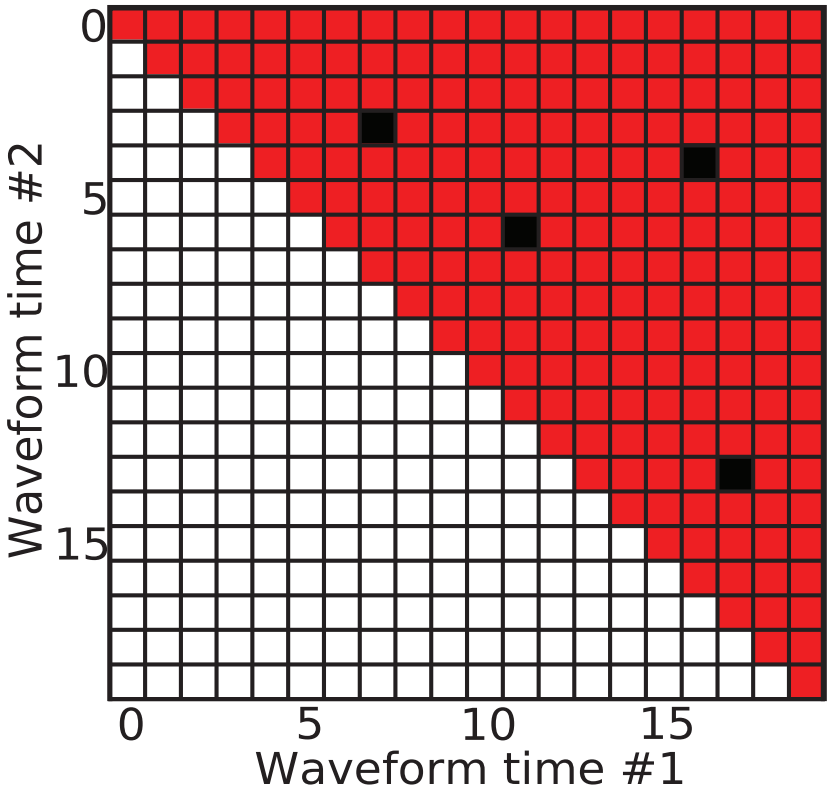


Searching for Similar Waveforms

Autocorrelation

Brute force search

Quadratic runtime scaling: $O(N^2)$



FAST

Efficient search

Near-linear runtime scaling: O(N)