Lesson 1-2: The SAC data format

i. Philosophy and structure

ii. Converting to, within, and from SAC format

Lesson 1-3: SAC processing philosophy

SAC has a complete set of reliable and well-tested processing elements to document, view, process, transform, and save data and results.

Given this abundant set of elements, how does one typically process seismic data?

This lecture will describe general approaches to getting the desired information from a seismogram collection.

Processing philosophy

- Organize – standardize/prepare data for analysis
- Interact – selection portion of waveform to operate on
- Process – calculate property / estimate parameter / transform data
- Display – graphically show result of processing to verify / validate

Organize

- Standardize file names
  - e.g. for vertical component data from station XYZ, want all data in files called XYZ.aabbcc.BHZ
  - best done with shell scripts of Unix commands
  - group three component data (ENZ, 123) as XYZ.aabbcc.BHE, XYZ.aabbcc.BHN, XYZ.aabbcc.BHZ

Organize

- Add relevant event information (origin time, lat, lon, depth, magnitude)
- Add component orientation if mis-oriented or not standard
- Cut data to common time window and length or merge fragments to usable length
- Deglitch (now rare but historical data has them)
- Decimate/interpolate to common sample rate

Organize (cont.)

- Remove instrument response / standardize to common response
- Remove data mean, trend
- Filter appropriately (low, high, or bandpass)
- Add travel time picks
- Preview and winnow bad data traces ...
- Write as SAC binary files; uniform format, uniform structure, different data.

Interact

- Generally graphical interaction with data (though need not be). Graphical interactions are quick and less error prone than typing; preferred.

- Types of interactions:
  - Data QC (quality control) – discard noisy/ unsuitable traces
  - Pick time window for analysis
  - Pick signal onset

Interact (cont.)

- Interaction graphical, but desired information is numeric.

- Save graphical pick results as numbers (usually time offsets) for processing step. Choices:
  - in memory
  - in the trace header information (will be saved as in SAC file with data)
  - external file

- For QC use, save either good file names, bad file names (or both)

Process

- Heart of analysis process.
- Usually specialized external program that SAC file is passed. Good programs:
  - produce text output (for debug or info purposes)
  - produce graphical output (for validation).
- Processing might also use SAC internal analysis elements (spectrum estimation, stacking, waveform picking, trace sample arithmetic).

Display

- Best confirmation of proper behavior of processing step is graphical output for validation / results.
- Display should follow processing step
- External program, if used, must produce a SAC output file that can be read by SAC for display

Useful commands for processing steps

- Organizing data
  - Change data in file headers: `CHNHDR`
  - Cut/merge data to common length: `CUT, CUTIM, MERGE`
  - Common time base: `CHNHDR ALLT, SYNCHRONIZE`
  - Deglitch: `DEGLITCH`
  - Remove mean and trend: `RMEAN, RTREND`
  - Tapering: `TAPER`
  - Filter: `LOWPASS, HIGHPASS, BANDPASS`
  - Instrument response: `TRANSFER`

Useful commands for processing steps (cont.)

- Organizing data (more)
  - Add traveltime picks: `TRAVELTIME` (SSS)
  - Preview and winnow: `PLOT1`, `MESSAGE`, `SETBB` with `(REPLY ...)`
  - Writing data: `WRITE`

- Interacting with user
  - Graphical interactions: `PLOTPK`, `PLOTRECORDSECTION / PRS` (SSS)
  - Text interactions: `MESSAGE`, `SETBB` with `(REPLY …)`

Useful commands for processing steps (cont.)

- Process
  - Invoke Unix command/external program: `SYSTEMCOMMAND, $RUN … $ENDRUN`
  - SAC processing facilities: SPE subprocess, SSS subprocess, `APK`

- Display
  - Graphical interactions: `PLOTPK, PRS` (SSS)
  - Text interactions: `MESSAGE, SETBB` with `(REPLY …)`

More information about SAC commands

- Start with
  - `SAC> help commands`
    - List of all commands (overwhelming)
  - `SAC> help apropos XXXX`
    - List of commands with XXXX in their description
- Then
  - `SAC> help YYYY`
    - Help for command YYYY
- Use web search engines
  - `http://www.iris.edu/software/sac/commands/func_commands.html`

Philosophy and structure

- SAC file: one file, one trace
- Trace consists of a fixed-length header, and variable length of data (determined by header information)
- Binary and character (alphanumeric) formats
- Data is either numerical (integer and real), logical (true/false), categorical (explosion source, nuclear source, earthquake source), or character (station code, event ID, arrival name)

Philosophy and structure (cont.)

- Alphanumeric data format intended for ease of writing and transfer between machine types
  - not widely used for processing due to need to convert to binary form for SAC operations
  - header data organized into sections, each section subdivided into lines
  - alphanumeric is data is text (.txt), not RTF or Word files (NOT .rtf or .doc)

Philosophy and structure (cont.)

- Binary format for efficient reading and writing
- Binary information in header and data portions of file: signed 32 bit integers, IEEE floating point real numbers
  - HUGE historical mistake in defining header format: no way to determine whether big-endian (Sun SPARC/Power PC) or little-endian (DEC/Intel) format
  - leads to MANY errors in user-written SAC file processing programs
  - (will teach strategies to avoid them)

Converting to and from SAC format

- SAC alphanumeric to binary
  - `SAC> read alpha` afile
  - `SAC> write binary` bfile
- SAC binary to alphanumeric
  - `SAC> read binary` bfile
  - `SAC> write alpha` afile

Converting to and from SAC format (cont.)

- GSE (Group of Scientific Experts) AutoDRM format to and from SAC
  - To SAC:
    - `SAC> readgse` gfile
    - `SAC> write` sfile
  - From SAC
    - `SAC> read` sfile
    - `SAC> writegse` gfile
  - Some information is lost in translation (multiple event origins, picks >10 in number)

Converting to and from SAC format (cont.)

- One way conversions available (convert other format to SAC):
  - CSS (Center for Seismic Studies)
  - MSEED (mini-SEED IRIS/PASSCAL format)
  - GCF (Guralp Compressed Format)
- Read in, then view/process/write as SAC files (or discard)
- Converting CSS data (separate command)
  - `SAC> readcss` cfile
- Converting MSEED data (variant of READ command)
  - `SAC> read mseed` mfile
- Converting GCF data (variant of READ command)
  - `SAC> read GCF` gfile

Converting to and from SAC format: Binary format and endian-ness

- Binary data formats have endian considerations to be aware of: storage order of binary data
  - Endian example: Arabic numerals
    - The number 137 is represented as
      - $1x10^2 + 3x10^1 + 7x10^0$
      - big-endian writing order is 1 3 7 (largest base-10 multiple first)
      - little-endian writing order is 7 3 1 (smallest base-10 multiple first; not used with arabic numbers, but with some linguistic representations of numbers, e.g. German “24” is “vier und swanzig” literally 4 and 20)

Converting to and from SAC format (cont.)

- Binary data grouped in 8-bit quantities called bytes (base 256) represented as 2 hexadecimal (base 16) digits 0-9,A-F (A=10, B=11, … F=15)

- As 1 byte number, 137 is hexadecimal 89
  - $8x16^1 + 9x16^0$

- As 4 byte big-endian binary integer, 137 is
  - 00 00 00 89

- As 4 byte little-endian binary integer, 137 is
  - 89 00 00 00

Converting to and from SAC format using custom programs

- Need to know three things to properly encode/ decode binary numbers:
  - actual endianness of input data
  - desired endianness of output data
  - endianness of computer processing the data

- It is complicated! – confuses most people (even programmers)

- Computer endianness:
  - Sun/Power PC computers – big-endian
  - DEC/Intel computers – little-endian

Converting to and from SAC format using custom programs

- SAC binary file endianness
  - No established convention (another design mistake!)
  - SAC2000 will read any endianness SAC file and will write file with machine endianness
  - MacSAC reads and writes big-endian SAC files ONLY
    - Conversion tool (`sactosac`) provided to convert ANY file endianness to big-endian
      - `sactosac -m` bfile ...

Converting to and from SAC format using custom programs

- How to avoid endianness problems when writing your own tools for data conversion
  - Convert to alphanumeric format, then SAC
  - Use subroutine packages that are fully cognizant of endianness and convert data using them
    - SAC programming library provides these routines (in `sacio.a` library; see “help library” in SAC)