2-1

i. **The SAC command**
ii. Reading and writing data
iii. Plotting and windowing
iv. Picking travel-times
v. Header manipulation

SAC commands

- SAC commands are single verbs (e.g., read, write) or compound words (e.g., filterdesign)
- These are followed by the options required for their operation, if none are specified the last options specified are reused, or the defaults if the command hasn't been run in this SAC session
- Multiple commands can be chained on the same line with ';' character separators
- Commands (and options) can be abbreviated to shortened forms
- A list of commands is available by typing:

```
SAC> help commands
```

- Help for a individual commands is available, e.g.:

```
SAC> help bandpass
```

- Or to look for commands by keyword:

```
SAC> help apropos keyword
```

Command history

- You can scroll back through previous commands using the up/down arrow
- By default, this is just in the current session
- However, can turn on transcription:

```
SAC> transcript history file ~/.saccommands
```

writes commands to the specified file. This can be local or global. The contents of this file is available when SAC is restarted.

- The command history is lost if SAC is killed (rather than being closed cleanly)
- The transcript command can be more generally used to record processing steps, see

```
SAC> help transcript
```

2-1

i. The SAC command
ii. **Reading and writing data**
iii. Plotting and windowing
iv. Picking travel-times
v. Header manipulation

Reading data

- Basic command to read data is, rather unsurprisingly, read

```
SAC> help read

 SUMMARY:
 Reads data from SAC data files on disk into memory.

 SYNTAX:
 READ [options] [filelist | wild]
```

- Can read one or multiple traces
- Defaults to SAC binary format, other formats (including ASCII) available
- Defaults to replace any traces currently in memory

Reading data

- Files are specified by a list and/or wildcards
- Wild cards are standard UN*X ones:
  - * matches any string of characters including an empty string
  - ? matches any single (non-null) character
  - [A,B,C] matches any character in the list
- Filenames can include directories
- Traces are read in the order specified, after expansion of the wild cards

Reading data: examples

- Names can be wildcarded ….

```
SAC> r SWAV.BH?
SWAV.BHE SWAV.BHN SWAV.BHZ
```

- Or listed

```
SAC> r SWAV.BHE SWAV.BHN SWAV.BHZ
```

- ‘more’ option appends traces into memory (instead of replacing existing traces

```
SAC> r SWAV.BHE
SAC> r more SWAV.BHN
SAC> r more SWAV.BHZ
```

- Directory references can be relative or absolute

```
SAC> r /tmp/SWAV.BH?
SAC> r ../SWAV.BH?
```

Writing data

- The write command outputs current traces

```
SAC> help write

 SUMMARY:
 Writes data in memory to disk.

 SYNTAX:
 WRITE {options} {namingoptions}
```

- Naming options include specifying a file list (no wildcards)

```
SAC> r SWAV.BH?
SAC> w SWAV.BHE SWAV.BHN SWAV.BHZ
```

- Or using the most recent read file list

```
SAC> r SWAV.BH?
SAC> w over
```

```
SAC> r SWAV.BH?
SAC> w append .new
SWAV.BHE.new SWAV.BHN.new SWAV.BHZ.new
```

2-1

i. The SAC command
ii. Reading and writing data
iii. Plotting and windowing
iv. Picking travel-times
v. Header manipulation

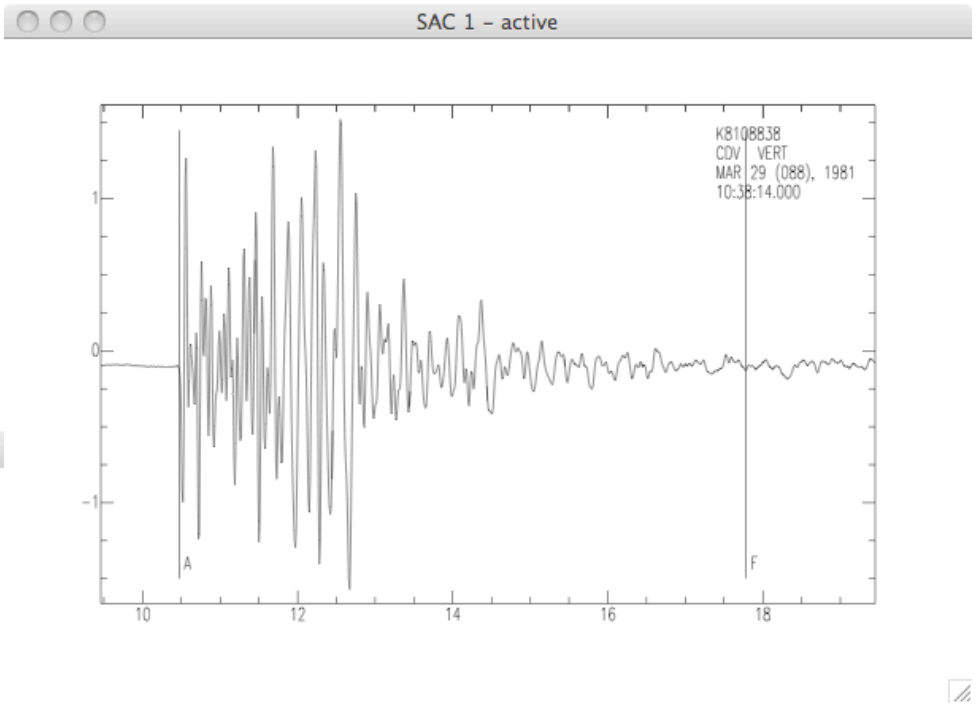
Plotting: window devices

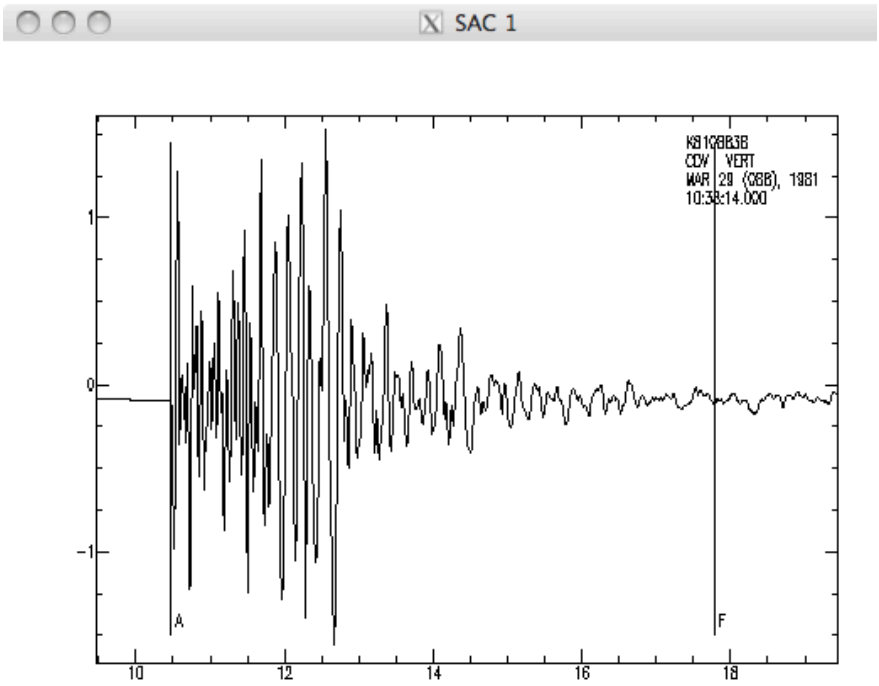
- SAC produces plots in one of several devices. These represent different OS windowing systems. Devices are started (or switched to) using the begindevice command.

```
SAC> help begindevices

 SUMMARY:
 Begins plotting to one or more graphics devices.

 SYNTAX:
 BEGINDEVICES devices
```

- Useful devices are:
  - Xwindows : the Xwindows library, available on most UN*X machines.
  - SGF : SAC's hardcopy device – plots are written to postscript-like files
  - Mac : a native Quartz viewer for MacOSX (MacSAC only)

Windowing examples





Plotting data

- The plot1 command plots each trace on a separate axis

```
SAC> r SWAV.BH?
SAC> plot1
```

- The plot2 command plots each trace on a common axis

```
SAC> r SWAV.BH?
SAC> plot2
```

- Many options exist for configuring size, labelling, positioning and multiple plot panels – see the help for details

Plotting options

- Time picks are also shown by default (the A, F, O, T0-T9 headers), labelled with the associated header string (KA, KF, KO, KT0-KT9)

- Plots can be paged :

```
SAC> plot1 perplot 5
```

- By default time scale is absolute, but traces can be plotted relative to their own reference times (useful when comparing traces from different times)

Plotting data

- The commands xlim and ylim set axis options (for next plot command
- Fixed axis limits:

```
SAC> xlim 0 10; p1
```

- Common axis limits (min/max of all traces):

```
SAC> ylim all; p1
```

- Restore defaults

```
SAC> ylim off; p1
```

- Note that for min/max, whole trace is taken, not just windowed region

Windowing data

- Often want to cut traces down to a window of interest (smaller, easier to plot and process)
- cut command sets a window of interest – subsequent read commands will only read in data in that window

```
SAC> cut 1000 1500; r ACKN.BH?
```

- Write commands will write out the windowed part of the trace only; be careful with overwriting original data with a shortened version
- Headers cannot be written to disk while cut is defined, so usual sequence is something like:

```
SAC> r ACKN.BH?
SAC> cut 1000 1500; r
SAC> w prepend SHORT_
SAC> cut off; r SHORT_ACKN.BH?
```

Producing permanent plots

- Default windowing device is X-windows, the other useful one is SGF (SAC Graphics Format)
- In order to produce a permanent plot:

```
SAC> begindevice sgf
SAC> p1
SAC> begindevice x
```

- This creates a file in the current working directory called ‘f001.sgf’
- Subsequent plot commands in the SGF device will produce f002.sgf, f003.sgf etc
- The counter is reset when SAC is restarted, so plots will be overwritten at this point
- SGF files can be translated to postscript (and thus on to PDF) using the utility sgftops:

```
$ sgftops f001.sgf myplot.ps
```

2-1

i. The SAC command
ii. Reading and writing data
iii. Plotting and windowing
iv. Picking travel-times
v. Header manipulation

Picking

- Pick travel times in data (usually) for futher analysis
- SAC can store picks in headers A, F and T0-T9
- Picking interface is started with the command plotpk (ppk)

```
SAC> help ppk

PLOTPK options
```

- This opens a data window with the current xlim/ylim options, with a cross hair. Keyboard shortcuts are used to control the picking, most important ones are:
  - o – resets the window to default limits
  - x – sets the first or second x limit to zoom to (i.e., pick min then max)
  - a and f set the A and F headers (often used to specify an analysis window)
  - t then 0-9 sets the corresponding header
  - q returns to the command prompt

Picking

- Picks are stored in the file headers *in memory*, the files on the disk are not automatically updated
- plotpk has a couple of options of note:
  - perplot *n* : only plot *n* traces per window (useful if you are picking many traces), use n and b keys to pages forwards and backwards
  - markall : apply picks to all traces in window

2-1

i. The SAC command
ii. Reading and writing data
iii. Plotting and windowing
iv. Picking travel-times
v. Header manipulation

Headers

- Headers store metadata about the trace
- These relate to:
  - the trace itself (NPTS, DELTA)
  - the reference time of the trace (NZYEAR, NZJDAY, …) and the trace's relation to it (B,E)
  - the 'event' (O, EVDP, EVLA, EVLO, KEVNM)
  - the station (STLA, STLO, STEL, KSTNM)
  - the raypath (AZ, BAZ, GCARC, DIST)

- Headers are set manually, or automatically computed by SAC based on other headers (for example, AZ, BAZ and GCARC are calculated when STLA, STLO, EVLA and EVLO are set)
- Headers can have a NULL value, in which case they are defined as unset

Listing headers

- The command listhdr (lh) lists headers for the current files in memory

```
SAC> help lh

 SUMMARY:
 Lists the values of selected header fields.

 SYNTAX:
 LISTHDR {listops} {hdrlist}
```

- By default, this lists all (non-null) headers for all files
- Can specify headers to list:

```
SAC> lh kzdate kztime


  FILE: 200496_2124_ACKN.BHE
  --------------------------
    kzdate = APR 05 (096), 2004
    kztime = 21:24:04.000
```

Extracting header information in UN*X

- The utility sacgethdr lists specified headers for one or more SAC files

```
$ sacgethdr stla,stlo *.BHZ
 ACKN.BHZ   64.9915 -110.8708
 BOXN.BHZ   63.8521 -109.7169
 CAMN.BHZ   63.7321 -110.8989
 COWN.BHZ   65.2680 -111.1860
 DSMN.BHZ   63.1799 -113.9004
 DVKN.BHZ   64.5092 -110.3096
 EKTN.BHZ   64.6985 -110.6097
 ...
```

- This can be very useful for scripting operations – this will be discussed in a later lesson

Setting headers

- The command chnhdr (ch) changes specified headers for the current files in memory

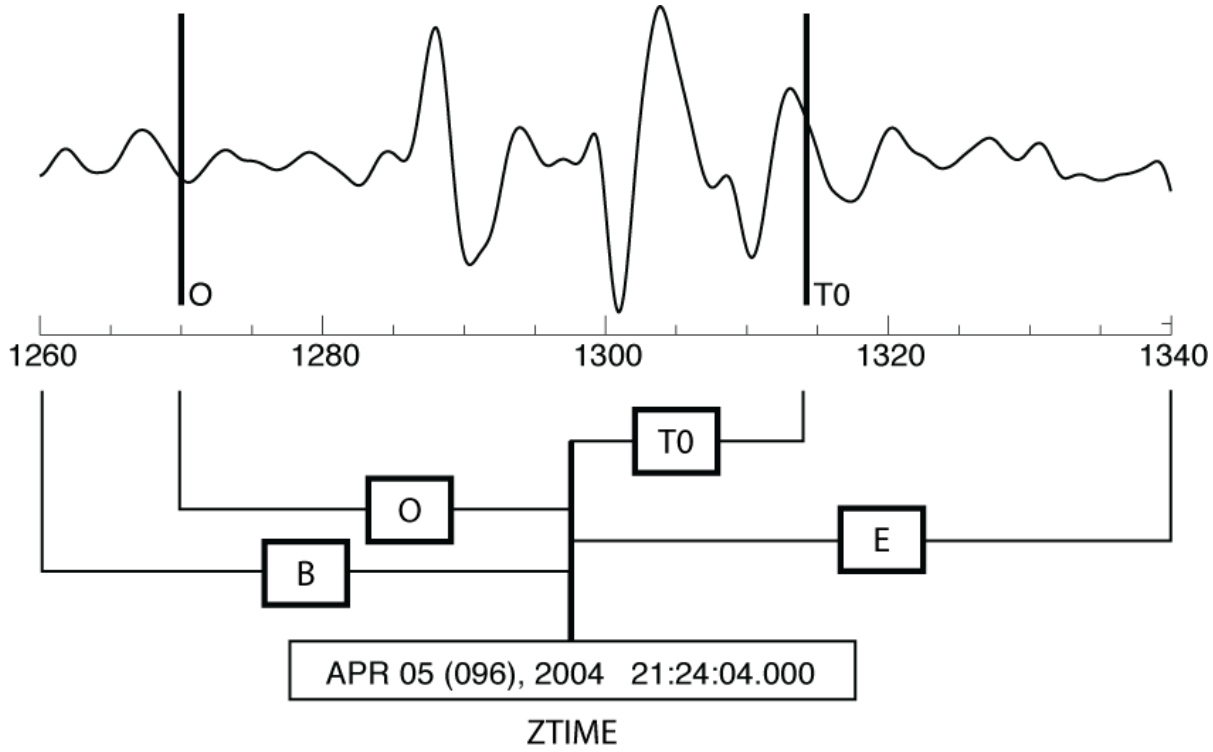
```
SAC> help ch

 SUMMARY:
 Changes the values of selected header fields.

 SYNTAX:
 CHNHDR  { file n1 n2 ... } field v  {field v ... }
```

- This changes the headers specified by *field* to values specified by *v* by default to all files in memory
- E.g., setting event information:

```
SAC> chnhdr evlo 70.840 evla 36.520 evdp 183.0
```

Time headers

- Trace reference time (the ZERO time) is stored in the header variables NZYEAR, NZJDAY, NZHOUR, NZMIN, NZSEC, NZMSEC
- Other time headers (B,E,A,F,O,T0-9) are stored relative to this:



Time headers

- Date is set by year and Julian day (i.e., day number in the year)
- Day number can be found with the UNIX command cal

```
$ cal 09 2010; cal -j 09 2010
   September 2010
Su Mo Tu We Th Fr Sa
           1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30

      September 2010
 Su  Mo  Tu  We  Th  Fr  Sa
             244 245 246 247
248 249 250 251 252 253 254
255 256 257 258 259 260 261
262 263 264 265 266 267 268
269 270 271 272 273
```

Setting time headers

- Time headers can be set relatively or absolutely (though they are always stored relatively)

```
SAC> chnhdr O -45
```

```
SAC> chnhdr O GMT 2006 075 13 45 28 600
```

- Time headers can be modified simultaneously:

```
SAC> chnhdr allt 415.045
```

- This shifts the reference time by a specified amount, and changes all the relative times accordingly
- The command synchronize makes the reference time of all traces in memory be the same (and updates relative times)

Writing headers

- Headers are set in memory, the file on the disk is unchanged
- To update headers in the disk file(s), use command writeheader (wh)

```
SAC> chnhdr O GMT 2006 075 13 45 28 600; wh
```

- Headers will also be written with the normal write commands

Example, setting up station and event information

```
SAC> r ACKN.BHN
SAC> chnhdr stla 64.9915 stlo -110.871; wh
SAC> chnhdr evla 36.520 evlo 70.840 evdp 183.0
SAC> chnhdr O GMT 2004 096 21 24 04 000; wh
SAC> lh o


  FILE: ACKN.BHN
  --------------
    o = -415.0

SAC> chnhdr allt 415; wh
```

- This loads a file, sets up the station and event headers
- Times are then shifted to be relative to the event time