# CHAPTER ONE

# Introduction

## 1.1 WHAT IS SAC?

SAC is an acronym for the Seismic Analysis Code, a command line tool for basic operations on time series data, especially seismic data. SAC includes a graphical interface for viewing and picking waveforms. It defines a standard type of file for storing and retrieving time series and also reads files written in other data formats (SEG-Y, MSEED, GCF) used during field collection of seismic data. SAC is self-contained and does not rely on network access for any of its capabilities, including documentation, which makes it useful for field data quality control.

SAC reads data formats (CSS and GSE formats) used by nuclear test monitoring agencies. It also contains programming language constructs that provide basic methods for developing elaborate, multi-step analysis methodologies. Collectively, these features make SAC a useful interactive platform upon which customized analytical methods may be built and prototypical procedures may be developed.

SAC is widely known. The IRIS Data Management Center (DMC), one of the largest whole-Earth seismological data repositories in existence, allows data to be requested in SAC form. The instrument response information provided by the DMC's SEED reading program, *rseed*, is usable by SAC in pole-zero or *evalresp* form. Owing to SAC's longevity, a rather large and well debugged software tool ecosystem has evolved around its file format. One such tool, *jweed*, searches for and retrieves data held by the DMC. SAC data and SAC-compatible instrument response information are among its output options. This and many other programs developed by individual researchers make SAC a natural choice for time series data analysis.

## 1.2 HISTORY AND DEVELOPMENT

SAC initially was developed in the early 1980s by the Livermore and Los Alamos National Laboratories in the Treaty Verification Program group. The developers were led by W. C. Tapley and Joe Tull, and the package incorporated parts of Dave Harris' XAP program (Harris, 1990). SAC's Fortran source code was distributed to interested academics as it became a tried, tested and valuable system for non-commercial seismic data processing. In this early distribution epoch, there was a collegial agreement between users and maintainers to send bug fixes and improvements to the developers in exchange for use. In 1986, SAC came to GH's notice, who used it in his thesis work. By about 1990, SAC had become the *de facto* standard analysis system for academic whole-Earth seismologists worldwide.

Beginning in about 1992, development of SAC was increasingly taken over by Livermore and access to the source code became restricted through distribution agreements with the lab. This culminated in the last source code release, version 10.6f, around 2003.

During this period, SAC's Fortran code base was converted to C language using an automatic Fortran to C conversion tool called *f2c*. This was apparently done in the belief that Fortran was too restrictive a language and that it hampered further development of SAC's features. Livermore continued SAC development using the C code base with a view to future commercial sales of a seismic data analysis product called SAC2000. In this version, SAC's functionality was to be extended by adding, among other features (Vergino and Snoke, 1993), a processing log database that would record all steps in transforming a trace from a raw form into a processed form. The design allowed for the processing steps to be tentatively saved and then either committed to memory or rolled back to an earlier state. During this period, distribution of SAC's source code ceased due to ongoing development and commercial licensing restrictions.

In about 1998, the IRIS Consortium recognized that SAC's core user community, essentially IRIS' membership, had no guarantee of affordable access to SAC's source code. IRIS began negotiation with Livermore to develop two strands, one with database features for use by nuclear monitoring organizations and another without database features for academic use. The commercialization efforts focused on the database-enabled version. In 2005, IRIS took up support and development of SAC2000 without the database features, leading to a SAC version called, in this book, SAC/IRIS. There was no academic community interest in the commercial SAC release, and Livermore's support for SAC2000 was eventually withdrawn.

During this time, the Fortran 10.6d code base (later integrated with 10.6f) continued to be maintained and developed at the University of Bristol. The bugs that existed in the Fortran code base were gradually eliminated (though they continued to exist in the C source version) and SAC's functionality continued to expand, particularly in the area of array processing of interest to Bristol researchers. This is the version documented in this book.

## 1.3 ALTERNATIVES TO SAC

**gSAC** is a name inspired by the GNU Project's free versions of the C compiler (gcc) and the Fortran compiler (gfortran). gSAC was developed by Bob Herrmann at Saint Louis University in response to SAC's monolithic internal structure and its previously closed source distribution, using advances in computer platforms. Over a period of about six weeks in 2004, gSAC re-implemented SAC's basic seismic trace manipulation functionality from scratch. Now gSAC is a group effort that provides documented tools for manipulating seismic traces which happen to be stored in SAC's file format.

**Seismic-Handler** was developed at the Seismological Observatory Gräfenberg by Klaus Stammler and several contributors. It is a software package that defines a set of waveform modification programs on a common data format and a scripting language to string together the programs to produce an analysis stream. There is an interactive graphical user interface for observatory purposes (e.g., daily routine seismicity analysis) and a command line version for scientific research. In 2008 the Deutsche Forschungsgemeinschaft (German Research Foundation) funded a project for further development of Seismic-Handler.

**SEISAN** is a package similar in structure to Seismic-Handler but oriented for use by observatories involved in routine seismic analysis. The system comprises a complete set of programs and a simple database for analyzing earthquakes from analog and digital data. SEISAN includes graphical user interaction facilities on waveform data to locate events, to edit events, to determine spectral parameters, seismic moment, and azimuth of arrival from three-component stations and to plot epicenters. A database search functionality exists to extract and operate on the data for particular events. Most of the programs can operate both conventionally (using a single file with many events) or on a database. SEISAN contains integrated research-type programs like coda Q, synthetic modeling and a complete system for seismic hazard calculation. The system is freely available for all non-commercial use. SEISAN was developed at the University of Bergen by Jens Havskov and Lars Ottemöller.

**SU/Seismic UNIX** is an open source seismic utilities package supported by the Center for Wave Phenomena at the Colorado School of Mines. The package provides an instant seismic research and processing environment for users running UNIX or UNIX-like operating systems. The package is a set of independent programs that exchange data in a common format through pipes. It has no graphical interface. The original developers were Stockwell and Cohen, though now it is an open source project with many contributors.

**AH** is a UNIX-inspired set of basic seismic operations (reading, filtering, decimation, etc.) on an input stream to transform data that is then delivered onto an output stream. It is based on UNIX pipes. The system was developed in the early 1990s by Dean Witte and Tom Boyd of Lamont-Doherty/Columbia University. The C language source code is still available online, although the package is no longer actively maintained.

**PITSA** was written by Frank Scherbaum (then at the University of Munich, now at the University of Potsdam) in the early 1990s and runs on IBM PCs and Sun workstations. It offers utilities for simple trace manipulation, like shifting or scaling of traces, adding or concatenating traces, stacking and others. Internally, PITSA uses a data format geared toward earthquake seismology, and it also reads plain ASCII and SEED files. PITSA's graphical user interaction is based on menus, utilizing dialog boxes and pop-up menus. PITSA no longer seems to be maintained, but its source code is available.

**MATLAB** offers a time series toolbox.

**R** is a free, open source system with built-in graphics oriented toward statistical analysis. It includes a time series library, and loadable libraries exist to read and write SAC files.

**DIY tools**. SAC's documented file structure has a useful collection of library routines usable from C, Fortran and Python that make it easy to develop personal tools for analyzing SAC time series. See Chapter 6 for further information.

## 1.4 SAC VARIANTS

**Fortran SAC.** This is SAC as implemented in Fortran source code. It was distributed by its developers up to version 10.6f. Source code for this version was also distributed in restricted form in some versions of the IASPEI Software Library circa 2003.

**SAC2000.** This is SAC translated from Fortran into C source code and subsequently maintained in C. Database capabilities were the principal development addition to this version and some new commands were also implemented. This version of SAC is no longer distributed.

**SAC/IRIS.** This is derived from SAC2000, without the database capabilities. It is actively maintained by the SAC development team under the aegis of the IRIS Consortium and is distributed by IRIS.

**MacSAC (SAC/BRIS).** This variant is derived from the 10.6d Fortran source code distribution and represents a superset of the capabilities of SAC/IRIS. The principal extensions relative to SAC/IRIS are in the capabilities of the macro language and significantly expanded handling capabilities for array data.

## 1.5 REQUIREMENTS AND INSTALLATION

**SAC/IRIS** is distributed through a licensing agreement with the IRIS Consortium. Distributions are available in source and binary forms from IRIS. Binary distributions are available for 32- and 64-bit Linux systems, 32- and 64-bit Macintosh systems and Solaris systems. Windows users must build from source in the Cygwin environment.

**MacSAC (SAC/BRIS)** is presently available in prepackaged distributions for MacOSX systems from 10.2 onward. The system automatically builds itself under MacOS, Solaris, FreeBSD and Linux systems from source releases based on 10.6d.

## 1.6 SCOPE OF THIS BOOK

The SAC command repertoire is vast – over 200 in all. We will not attempt to cover them all in this book. Our goal is rather to provide an introduction to SAC's basic concepts and its basic command set. Consequently, many commands are not included. An exhaustive list of commands with a keyword index of concepts derived from the command description appears in the appendix. This list, along with the help information built into SAC, will hopefully lead you to speculate on, find and use SAC's broader functionality.

The first five chapters of this book constitute basic material. The following chapters emphasize SAC features not adequately documented anywhere else. Our goal here is to supply that documentation and to show, with examples, the occasionally surprising utility of those features.

The final chapter is a description of how SAC may be used to implement some of the standard data analysis procedures used by seismologists on teleseismic data. Though the procedures are eminently serviceable as they stand, their inclusion here is to serve as examples of how SAC's capabilities may be used more effectively. They represent the distillation of over 20 years of experience with SAC and will reward study and reflection. Looking into the future, one can imagine developing procedures for analyzing station noise levels from long sequences of MSEED field data blockettes, for aspects of ambient noise study and for normal mode seismology.

Note that some capabilities provided by SAC require significant skill to use properly, particularly correcting for instrument response. Unfortunately, the subtleties of this topic are beyond the scope of a SAC-oriented text such as this one. Other problematic areas include attenuation correction and earthquake location. SAC provides features that either perform these functions or link with standard programs that do them. Again, the scope widens significantly when these topics are included, and so, with apologies, we have opted to omit them here. Sorry.

# CHAPTER TWO

# The SAC data format

## 2.1 PHILOSOPHY AND STRUCTURE

### SAC file format

A seismic data trace is a set of data points that is continuous in time but that may not have been sampled at an even rate. SAC's simple approach to seismic data deems that there is one seismic trace per file. Each file contains a header that describes the trace (also known as metadata) and a section that contains the actual data. The header occupies a fixed-length position at the beginning of each file, followed by the variable-length data section.

Header data is of mixed type: integer and real values, logical (true/false) values, categorical values (distinct properties like explosive source, nuclear source, earthquake source), or text (station code, event identification, wave arrival type). The seismic data in a trace is a sequence of real-valued numbers representing the sampled physical property.

### Alphanumeric and binary forms

There are both binary and alphanumeric (character) formats for a SAC file. The binary version is a more compact format that is efficiently read and written, while the alphanumeric format is easier for user programs to read and write.

The alphanumeric data format[1] is intended for ease of reading and writing and for transfer between different machine types. In this format, the header data is organized into

[1] Word processors should not be used to work with SAC alphanumeric data files. These are not in RTF or Word format, even though they contain text. The relevant format is TXT (in DOS terminology) and any editing software used to prepare or edit an alphanumeric file should be capable of writing this file type, which is devoid of typesetting formatting information.

sections based on the type of data, with each section subdivided into lines. While not an intuitive organization, this makes it easier to read and write the header data because all of the data on each line of the file is of the same type: integer, real, etc.

In contrast, the binary SAC format is compact and efficient to read and write. In this format, the header data is either integer or real floating point numbers (in IEEE 754 standard format). SAC binary data is always stored in single precision (32-bit) IEEE 754 floating point format.

### Interconversion of formats

Converting to and from SAC files of different format is easy with SAC. From alphanumeric to binary,

```
SAC> read alpha <afile>
SAC> write binary <bfile>
```

From binary to alphanumeric,

```
SAC> read binary <bfile>
SAC> write alpha <afile>
```

## 2.2 CONVERSION FROM OTHER DATA FORMATS

### GSE files

SAC is capable of reading files in other common data exchange formats. One common format generated by AutoDRM software (Kradolfer, 1996) is the GSE format (Group of Scientific Experts) (GSETT-3, 1995), in use by the United Nations' International Monitoring System, part of the Comprehensive Nuclear Test-Ban Treaty framework. This alphanumeric format can be directly read and written by SAC. To read GSE data into SAC and rewrite it in SAC format,

```
SAC> readgse <gfile>
SAC> write <sfile>
```

and to write GSE-format data when the original form is SAC,

```
SAC> read <sfile>
SAC> writegse <gfile>
```

Note that there is a separate SAC command to read and write GSE data; the `READ` command is not used.

Many traces can be present in one GSE file. The file usually is a string of messages produced by an AutoDRM and sometimes is the text of an e-mail. `READGSE` will skip the parts of the file that do not contain trace data and read the traces contained in the file into memory. The various traces may be selected for further processing by writing them as SAC files individually.

The conversion between GSE and SAC formats is not isomorphic. Some information is lost in the conversion from GSE into SAC format. Multiple origins for an event are lost; SAC only has a single origin associated with each trace. Arrival picks in a trace in excess of 10 are also lost. Any information loss is reported by SAC to the user, however.

### SEG Y, MSEED, GCF and CSS formats

Unlike the various SAC formats and the GSE format, the other formats cannot be written by SAC, only read by it.

#### *SEG Y*

The SEG Y data format is based on a standard developed by the Society of Exploration Geophysicists for storing geophysical data. The first definition of the format was in 1973 and it was documented in 1975 (Barry *et al.*, 1975). Defined when computers were larger than automobiles and when magnetic tape was the most portable recording medium, many variants emerged as different user communities modified it to versions suited to more modern storage media. The PASSCAL project defined one of these variants to handle field data gathered during passive seismic experiments. This is the version read by SAC. Briefly, it has no 3600-byte reel header, requires that the trace code header value be seismic (type code 1) and allows only one waveform per file.

To read SEG Y data, use a variant of SAC's READ command,

```
SAC> read segy <sfile>
```

which reads the trace from the designated SEG Y file.

#### *MSEED*

MSEED (mini-SEED) (IRIS, 2006) is a format defined by the International Federation of Digital Seismograph Networks principally for data archiving and exchange. The data format is also output by field dataloggers. Consequently, this makes SAC a useful tool for field computers for field data quality control.

To read MSEED data, use a variant of SAC's READ command,

```
SAC> read mseed <mfile>
```

MSEED data is commonly collected in short packets, each with a time stamp and a sequence of samples starting at that time. Consequently, the data might not be continuous across packet boundaries. Any data overlaps or gaps will be reported by SAC when read.

#### *GCF*

Another data format encountered in the field is GCF (Güralp Compressed Format)[2]. This is a raw data format output by dataloggers manufactured by Güralp Systems. To read GCF data, use a variant of SAC's READ command,

```
SAC> read GCF <gfile>
```

GCF data is collected in blocks of 1024 characters. Consequently, the data might not be continuous across block boundaries. As with MSEED, any data overlaps or gaps will be reported.

#### *CSS*

CSS format is a format defined by the Center for Seismic Studies and is a waveform database format used by the International Monitoring System, with separate files for trace metadata

[2] Defined in Güralp online information at `http://www.guralp.com/gcf-format/`

and trace data (Anderson *et al.*, 1990). Consequently, at least three separate files must be grouped to describe a trace. The grouped files are associated through one master waveform metadata file. The metadata files have names with the same prefix but different suffixes. Files with suffixes `.wfdisc` and `.origin` pertain to the waveform metadata and the origin information for the event, respectively. A third file, whose location is given in the `.wfdisc` file, contains the data for one or more traces. Each trace is described by one line in the `wfdisc` file, so one CSS metadata file can describe one or more traces.

To read a CSS file in SAC, provide the name of the `wfdisc` file, either with or without the suffix:

```
SAC> readcss <cssdata>
```

This implies that given a file prefix of `cssdat`, files called `cssdat.wfdisc` and `cssdat.origin` also exist. Alternatively, the full `wfdisc` file can be specified on the READCSS command, which is useful when pattern matching for a group of files.

Many traces can potentially be present in a CSS file collection. Using suitable READCSS command options, subsets of the traces may be chosen based on the channel name, station name and frequency range of the sensor. By default, READCSS reads all of them.

## 2.3 BYTE-ORDER ISSUES

Information in SAC's file header unfortunately does not include an explicit indication of the data's type of binary format. Contemporary processors organize data in memory in either most-significant byte order (big-endian; SPARC or PowerPC order) or least-significant byte order (little-endian; DEC/Intel order).

The added complexity in reading and writing data in the proper way may lead to many errors in user-written programs that read and write SAC files. One strategy to avoid byte-order problems is to read and write them in alphanumeric format. This has the additional advantage that the data is easily verified simply by viewing the file as text. Alphanumeric format is a useful intermediate form to convert data from another format not recognized by SAC to make it SAC-readable.

For programs that analyze or change data in a processing tool chain, there is more of an emphasis on performance. In this case, working with the binary format is more efficient. SAC provides a set of user-callable subroutines for easy access to header and data in SAC binary files. These routines handle the byte-order related problems automatically and can cope with past or future changes to the data format in SAC binary files (changes in the binary file format are being considered as of November 2012 and will probably occur in 2013 or 2014). The routines are provided in library form along with SAC. See Chapter 6 for further details about accessing SAC file information from programs and SAC's built-in help entry for its library of user-callable subroutines (`HELP LIBRARY`).

Unfortunately, SAC binary file byte order is not specified in any standard. Because SAC was originally developed on big-endian machines (Sun, Prime and Ridge workstations), this is the expected byte order, and any files written by SAC will be in big-endian order no matter what the underlying machine type is. Some versions of SAC (notably SAC/IRIS) will write binary files in the machine's native format. If one is encountered, SAC will report that the format is unexpected. A utility program (called `sactosac` and distributed with SAC) is available to switch to the proper byte order. See SAC's help information for its utility programs (`HELP UTILITIES`) for further usage information.

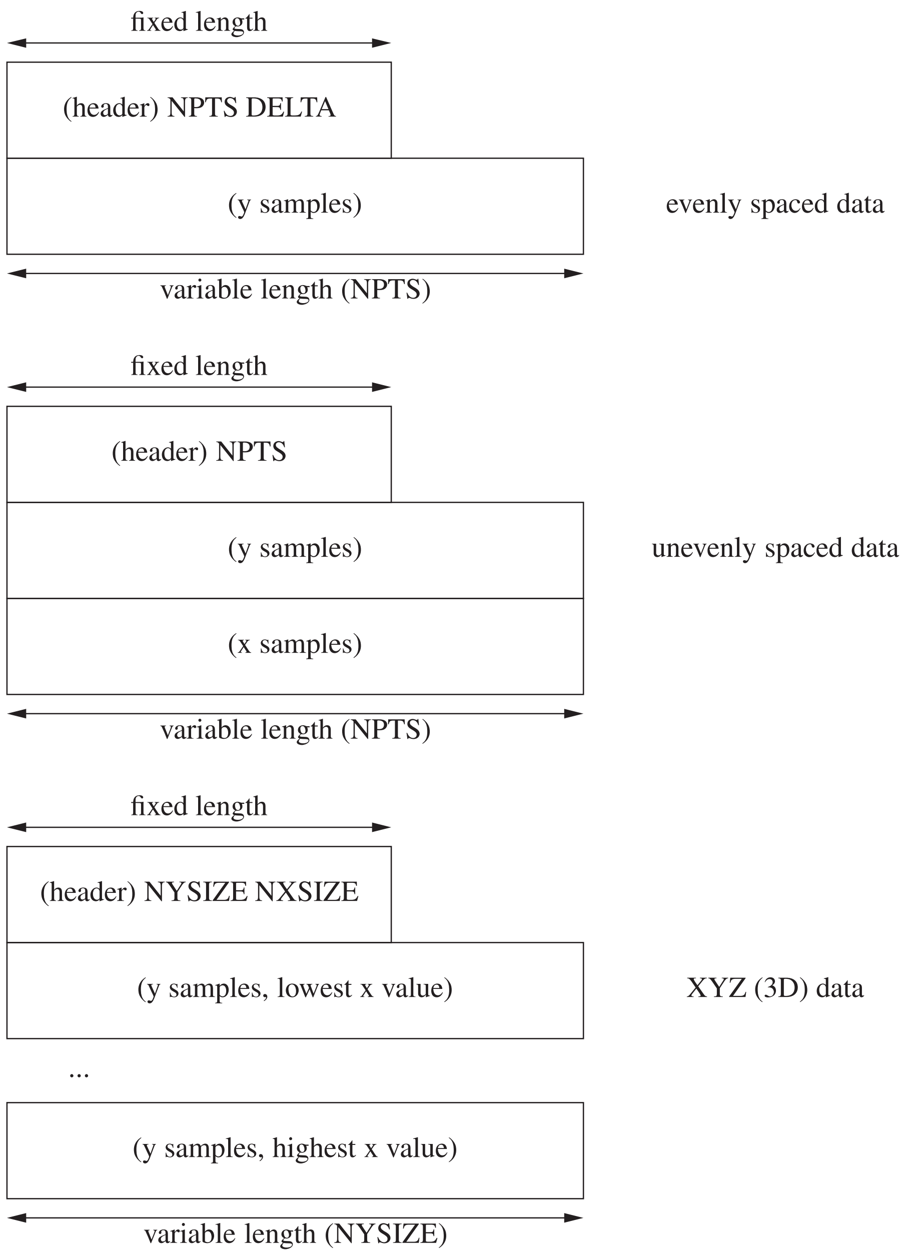

**Figure 2.1** Schematic layout of SAC files of different types. All types have a fixed-length header and a variable-length data portion. For evenly spaced data, the `NPTS` entry in the file header specifies the number of data points (sampled every `DELTA` seconds). For unevenly spaced data, `NPTS` in the header specifies the number of samples, but the sampled values appear first in the data section followed by the time at which the sample was taken. For XYZ, or 3D data (see Chapter 10), the sampled values in the Y direction at the lowest X value appear first, followed by samples for the next-higher X up to `NXSIZE` X values. `NYSIZE` gives the Y dimension.

## 2.4 SAC FILE LAYOUT

SAC provides essentially three types of data files. The metadata-containing header is fixed length. Depending on the number of data points and the type of the file indicated in the header, a variable number of data values will follow in the file. How many data values are present depends on the type of the file. Simple time series (the most common type – IFTYPE of ITIME) have a fixed number of samples (NPTS) that are separated by a fixed sample interval (DELTA). In contrast, unevenly sampled data files (IFTYPE of IXY) have a fixed number of samples separated by random time intervals. Consequently, DELTA has no significance, and the sample values are followed by the individual times at which the samples were made. The third type of file is XYZ (or 3D data – IFTYPE of IXYZ), where samples are made on a fixed spatial grid of (X,Y) positions. In this case, the header provides the number of X and Y sample positions (NXSIZE and NYSIZE), and sample values follow for each of the NXSIZE $\times$ NYSIZE points. The ideas are sketched in Figure 2.1.