

**NAME**

SoX – Sound eXchange, the Swiss Army knife of audio manipulation

**SYNOPSIS**

```
sox [global-options] [format-options] infile1
    [[format-options] infile2] ... [format-options] outfile
    [effect [effect-options]] ...
```

```
play [global-options] [format-options] infile1
    [[format-options] infile2] ... [format-options]
    [effect [effect-options]] ...
```

```
rec [global-options] [format-options] outfile
    [effect [effect-options]] ...
```

**DESCRIPTION****Introduction**

SoX reads and writes audio files in most popular formats and can optionally apply effects to them; it can combine multiple input sources, synthesise audio, and, on many systems, act as a general purpose audio player or a multi-track audio recorder. It also has limited ability to split the input in to multiple output files.

Almost all SoX functionality is available using just the **sox** command, however, to simplify playing and recording audio, if SoX is invoked as **play** the output file is automatically set to be the default sound device and if invoked as **rec** the default sound device is used as an input source. Additionally, the **soxi**(1) command provides a convenient way to just query audio file header information.

The heart of SoX is a library called libSoX. Those interested in extending SoX or using it in other programs should refer to the libSoX manual page: **libsox**(3).

SoX is a command-line audio processing tool, particularly suited to making quick, simple edits and to batch processing. If you need an interactive, graphical audio editor, use **audacity**(1).

\* \* \*

The overall SoX processing chain can be summarised as follows:

Input(s) → Combiner → Effects → Output(s)

To show how this works in practise, here is a selection of examples of how SoX might be used. The simple

```
sox recital.au recital.wav
```

translates an audio file in Sun AU format to a Microsoft WAV file, whilst

```
sox recital.au -r 12k -b 8 -c 1 recital.wav vol 0.7 dither
```

performs the same format translation, but also changes the audio sampling rate & sample size, down-mixes to mono, and applies the **vol** and **dither** effects.

```
sox -r 8k -u -b 8 -c 1 voice-memo.raw voice-memo.wav
```

converts ‘raw’ (a.k.a. ‘headerless’) audio to a self-describing file format,

```
sox slow.aiff fixed.aiff speed 1.027
```

adjusts audio speed,

```
sox short.au long.au longer.au
```

concatenates two audio files, and

```
sox -m music.mp3 voice.wav mixed.flac
```

mixes together two audio files.

```
play "The Moonbeams/Greatest/*.ogg" bass +3
```

plays a collection of audio files whilst applying a bass boosting effect,

```
play -n -c1 synth sin %-12 sin %-9 sin %-5 sin %-2 fade q 0.1 1 0.1
```

plays a synthesised ‘A minor seventh’ chord with a pipe-organ sound,

```
rec -c 2 test.aiff trim 0 10
```

records 10 seconds of stereo audio, and

```
rec -M take1.aiff take1-dub.aiff
```

records a new track in a multi-track recording.

```
rec -r 44100 -2 -s -p silence 1 0.50 0.1% 1 10:00 0.1% | \
  sox -p song.ogg silence 1 0.50 0.1% 1 2.0 0.1% : \
  newfile : restart
```

records a stream of audio such as LP/cassette and splits in to multiple audio files at points with 2 seconds of silence. Also does not start recording until it detects audio is playing and stops after it sees 10 minutes of silence.

N.B. Detailed explanations of how to use *all* SoX parameters, file formats, and effects can be found below in this manual, and in **soxformat(7)**.

## File Format Types

There are two types of audio file format that SoX can work with. The first is ‘self-describing’; these formats include a header that completely describes the characteristics of the audio data that follows. The second type is ‘headerless’ (or ‘raw data’); here, the audio data characteristics must be described using the SoX command line.

The following four characteristics are sufficient to describe the format of audio data such that it can be processed with SoX:

### sample rate

The sample rate in samples per second (‘Hertz’ or ‘Hz’). For example, digital telephony traditionally uses a sample rate of 8000 Hz (8 kHz); audio Compact Discs use 44100 Hz (44.1 kHz); Digital Audio Tape and many computer systems use 48 kHz; professional audio systems typically use 96 or 192 kHz.

### sample size

The number of bits used to store each sample. The most popular is 16-bit (two bytes); 8-bit (one byte) was popular in the early days of computer audio, and is still used in telephony; 24-bit (three bytes) is used, primarily as an intermediate format, in the professional audio arena. Other sizes are also used.

### data encoding

The way in which each audio sample is represented (or ‘encoded’). Some encodings have variants with different byte-orderings or bit-orderings; some ‘compress’ the audio data, i.e. the stored audio data takes up less space (i.e. disk-space or transmission band-width) than the other format parameters and the number of samples would imply. Commonly-used encoding types include floating-point,  $\mu$ -law, ADPCM, signed-integer PCM, and FLAC.

### channels

The number of audio channels contained in the file. One (‘mono’) and two (‘stereo’) are widely used. ‘Surround sound’ audio typically contains six or more channels.

The term ‘bit-rate’ is sometimes used as an overall measure of an audio format and may incorporate elements of all of the above.

Most self-describing formats also allow textual ‘comments’ to be embedded in the file that can be used to describe the audio in some way, e.g. for music, the title, the author, etc.

One important use of audio file comments is to convey ‘Replay Gain’ information. SoX supports applying

Replay Gain information, but not generating it. Note that by default, SoX copies input file comments to output files that support comments, so output files may contain Replay Gain information if some was present in the input file. In this case, if anything other than a simple format conversion was performed then the output file Replay Gain information is likely to be incorrect and so should be recalculated using a tool that supports this (not SoX).

The **soxi**(1) command can be used to display information from audio file headers.

### Determining & Setting The File Format

There are several mechanisms available for SoX to use to determine or set the format characteristics of an audio file. Depending on the circumstances, individual characteristics may be determined or set using different mechanisms.

To determine the format of an input file, SoX will use, in order of precedence and as given or available:

1. Command-line format options.
2. The contents of the file header.
3. The filename extension.

To set the output file format, SoX will use, in order of precedence and as given or available:

1. Command-line format options.
2. The filename extension.
3. The input file format characteristics, or the closest to them that is supported by the output file type.

For all files, SoX will exit with an error if the file type cannot be determined; command-line format options may need to be added or changed to resolve the problem.

### Play, Rec, & Default Audio Devices

Some systems provide more than one type of (SoX-compatible) audio driver, e.g. ALSA & OSS, or SUNAU & AO. Systems can also have more than one audio device (a.k.a. ‘sound card’). If more than one audio driver has been built-in to SoX, and the default selected by SoX when using **rec** or **play** is not the one that is wanted, then the **AUDIODRIVER** environment variable can be used to override the default. For example (on many systems):

```
set AUDIODRIVER=oss
play ...
```

For **rec**, **play**, and **sox**, the **AUDIODEV** environment variable can be used to override the default audio device; e.g.

```
set AUDIODEV=/dev/dsp2
play ...
sox ... -t oss
```

or

```
set AUDIODEV=hw:0
play ...
sox ... -t alsa
```

(Note that the syntax of the **set** command may vary from system to system.)

When playing a file with a sample rate that is not supported by the audio output device, SoX will automatically invoke the **rate** effect to perform the necessary sample rate conversion. For compatibility with old hardware, here, the default **rate** quality level is set to ‘low’; however, this can be changed if desired, by explicitly specifying the **rate** effect with a different quality level, e.g.

```
play ... rate -m
```

or by setting the environment variable **PLAY\_RATE\_ARG** to the desired quality option, e.g.

```
set PLAY_RATE_ARG=-m
```

```
play ...
```

(Note that the syntax of the **set** command may vary from system to system.)

To help with setting a suitable recording level, SoX includes a simple VU meter which can be invoked (before making the actual recording) as follows:

```
rec -n
```

The recording level should be adjusted (using the system-provided mixer program, not SoX) so that the meter is *at most occasionally* full scale, and never ‘in the red’ (an exclamation mark is shown).

### Accuracy

Many file formats that compress audio discard some of the audio signal information whilst doing so; converting to such a format then converting back again will not produce an exact copy of the original audio. This is the case for many formats used in telephony (e.g. A-law, GSM) where low signal bandwidth is more important than high audio fidelity, and for many formats used in portable music players (e.g. MP3, Vorbis) where adequate fidelity can be retained even with the large compression ratios that are needed to make portable players practical.

Formats that discard audio signal information are called ‘lossy’, and formats that do not, ‘lossless’. The term ‘quality’ is used as a measure of how closely the original audio signal can be reproduced when using a lossy format.

Audio file conversion with SoX is lossless when it can be, i.e. when not using lossy compression, when not reducing the sampling rate or number of channels, and when the number of bits used in the destination format is not less than in the source format. E.g. converting from an 8-bit PCM format to a 16-bit PCM format is lossless but converting from an 8-bit PCM format to (8-bit) A-law isn’t.

**N.B.** SoX converts all audio files to an internal uncompressed format before performing any audio processing; this means that manipulating a file that is stored in a lossy format can cause further losses in audio fidelity. E.g. with

```
sox long.mp3 short.mp3 trim 10
```

SoX first decompresses the input MP3 file, then applies the **trim** effect, and finally creates the output MP3 file by recompressing the audio—with a possible reduction in fidelity above that which occurred when the input file was created. Hence, if what is ultimately desired is lossily compressed audio, it is highly recommended to perform all audio processing using lossless file formats and then convert to the lossy format only at the final stage.

**N.B.** Applying multiple effects with a single SoX invocation will, in general, produce more accurate results than those produced using multiple SoX invocations; hence this is also recommended.

### Clipping

Clipping is distortion that occurs when an audio signal level (or ‘volume’) exceeds the range of the chosen representation. It is nearly always undesirable and so should usually be corrected by adjusting the level prior to the point at which clipping occurs.

In SoX, clipping could occur, as you might expect, when using the **vol** effect to increase the audio volume, but could also occur with many other effects, when converting one format to another, and even when simply playing the audio.

Playing an audio file often involves re-sampling, and processing by analogue components that can introduce a small DC offset and/or amplification, all of which can produce distortion if the audio signal level was initially too close to the clipping point.

For these reasons, it is usual to make sure that an audio file’s signal level does not exceed around 70% of the maximum (linear) range available, as this will avoid the majority of clipping problems. SoX’s **stat** effect can assist in determining the signal level in an audio file; the **gain** or **vol** effect can be used to prevent clipping, e.g.

```
sox dull.au bright.au gain -6 treble +6
```

guarantees that the treble boost will not clip.

If clipping occurs at any point during processing, then SoX will display a warning message to that effect.

### Input File Combining

SoX's input combiner can be configured (see **OPTIONS** below) to combine multiple files using any of the following methods: 'concatenate', 'sequence', 'mix', 'mix-power', or 'merge'. The default method is 'sequence' for **play**, and 'concatenate' for **rec** and **sox**.

For all methods other than 'sequence', multiple input files must have the same sampling rate; if necessary, separate SoX invocations can be used to make sampling rate adjustments prior to combining.

If the 'concatenate' combining method is selected (usually, this will be by default) then the input files must also have the same number of channels. The audio from each input will be concatenated in the order given to form the output file.

The 'sequence' combining method is selected automatically for **play**. It is similar to 'concatenate' in that the audio from each input file is sent serially to the output file, however here the output file may be closed and reopened at the corresponding transition between input files—this may be just what is needed when sending different types of audio to an output device, but is not generally useful when the output is a normal file.

If either the 'mix' or 'mix-power' combining method is selected, then two or more input files must be given and will be mixed together to form the output file. The number of channels in each input file need not be the same, however, SoX will issue a warning if they are not and some channels in the output file will not contain audio from every input file. A mixed audio file cannot be un-mixed (without reference to the original input files).

If the 'merge' combining method is selected, then two or more input files must be given and will be merged together to form the output file. The number of channels in each input file need not be the same. A merged audio file comprises all of the channels from all of the input files; un-merging is possible using multiple invocations of SoX with the **remix** effect. For example, two mono files could be merged to form one stereo file; the first and second mono files would become the left and right channels of the stereo file.

When combining input files, SoX applies any specified effects (including, for example, the **vol** volume adjustment effect) after the audio has been combined; however, it is often useful to be able to set the volume of (i.e. 'balance') the inputs individually, before combining takes place.

For all combining methods, input file volume adjustments can be made manually using the **-v** option (below) which can be given for one or more input files; if it is given for only some of the input files then the others receive no volume adjustment. In some circumstances, automatic volume adjustments may be applied (see below).

The **-V** option (below) can be used to show the input file volume adjustments that have been selected (either manually or automatically).

There are some special considerations that need to be made when mixing input files:

Unlike the other methods, 'mix' combining has the potential to cause clipping in the combiner if no balancing is performed. So here, if manual volume adjustments are not given, to ensure that clipping does not occur, SoX will automatically adjust the volume (amplitude) of each input signal by a factor of  $1/n$ , where  $n$  is the number of input files. If this results in audio that is too quiet or otherwise unbalanced then the input file volumes can be set manually as described above; using the **norm** effect on the mix is another alternative.

If mixed audio seems loud enough at some points through the mixed audio but too quiet in others, then dynamic-range compression should be applied to correct this—see the **compand** effect.

With the 'mix-power' combine method, the mixed volume is appropriately equal to that of one of the input signals. This is achieved by balancing using a factor of  $1/\sqrt{n}$  instead of  $1/n$ . Note that this balancing factor does not guarantee that no clipping will occur, however, in many cases, the number of clips will be low and the resultant distortion imperceptible.

## Output Files

SoX's default behavior is to take one or more input files and write them to a single output file.

This behavior can be changed by specifying the pseudo-effect 'newfile' within the effects list. SoX will then enter multiple output mode.

In multiple output mode, a new file is created when the effects prior to the 'newfile' indicate they are done. The effects chain listed after 'newfile' is then started up and its output is saved to the new file.

In multiple output mode, a unique number will automatically be appended to the end of all filenames. If the filename has an extension then the number is inserted before the extension. This behavior can be customized by placing a %n anywhere in the filename where the number should be substituted. An optional number can be placed after the % to indicate a minimum fixed width for the number.

Multiple output mode is not very useful unless an effect that will stop the effects chain early is specified before the 'newfile'. If end of file is reached before the effects chain stops itself then no new file will be created as it would be empty.

The following is an example of splitting the first 60 seconds of an input file in to two 30 second files and ignoring the rest.

```
sox song.wav ringtone%n.wav trim 0 30 : newfile : trim 0 30
```

## Stopping SoX

Usually SoX will complete its processing and exit automatically once it has read all available audio data from the input files.

If desired, it can be terminated earlier by sending an interrupt signal to the process (usually by pressing the keyboard interrupt key which is usually Ctrl-C). This is a natural requirement in some circumstances, e.g. when using SoX to make a recording. Note that when using SoX to play multiple files, Ctrl-C behaves slightly differently: pressing it once causes SoX to skip to the next file; pressing it twice in quick succession causes SoX to exit.

Another option to stop processing early is to use an effect that has a time period or sample count to determine the stopping point. The trim effect is an example of this. Once all effects chains have stopped then SoX will also stop.

## FILENAMES

Filenames can be simple file names, absolute or relative path names, or URLs (input files only). Note that URL support requires that **wget**(1) is available.

Note: Giving SoX an input or output filename that is the same as a SoX effect-name will not work since SoX will treat it as an effect specification. The only work-around to this is to avoid such filenames; however, this is generally not difficult since most audio filenames have a filename 'extension', whilst effect-names do not.

## Special Filenames

The following special filenames may be used in certain circumstances in place of a normal filename on the command line:

- SoX can be used in simple pipeline operations by using the special filename '-' which, if used in place of an input filename, will cause SoX will read audio data from 'standard input' (stdin), and which, if used in place of the output filename, will cause SoX will send audio data to 'standard output' (stdout). Note that when using this option, the file-type (see **-t** below) must also be given.

"|program [options] ..."

This can be used in place of an input filename to specify the the given program's standard output (stdout) be used as an input file. Unlike – (above), this can be used for several inputs to one SoX command. For example, if 'genw' generates mono WAV formatted signals to its standard output,

then the following command makes a stereo file from two generated signals:

```
sox -M -t wav " |genw --imd -" -t wav " |genw --thd -" out.wav
```

If **-t** is not given then the signal is assumed (and checked) to be in SoX's native **.sox** format (see **-p** below and **soxformat(7)**).

#### **-p, --sox-pipe**

This can be used in place of an output filename to specify that the SoX command should be used as in input pipe to another SoX command. For example, the command:

```
play " |sox -n -p synth 2" " |sox -n -p synth 2 tremolo 10" stat
```

plays two 'files' in succession, each with different effects.

**-p** is in fact an alias for '**-t sox -**'.

#### **-d, --default-device**

This can be used in place of an input or output filename to specify that the default audio device (if one has been built into SoX) is to be used. This is akin to invoking **rec** or **play** (as described above).

#### **-n, --null**

This can be used in place of an input or output filename to specify that a 'null file' is to be used. Note that here, 'null file' refers to a SoX-specific mechanism and is not related to any operating-system mechanism with a similar name.

Using a null file to input audio is equivalent to using a normal audio file that contains an infinite amount of silence, and as such is not generally useful unless used with an effect that specifies a finite time length (such as **trim** or **synth**).

Using a null file to output audio amounts to discarding the audio and is useful mainly with effects that produce information about the audio instead of affecting it (such as **noiseprof** or **stat**).

The sampling rate associated with a null file is by default 48 kHz, but, as with a normal file, this can be overridden if desired using command-line format options (see below).

### **Supported File & Audio Device Types**

See **soxformat(7)** for a list and description of the supported file formats and audio device drivers.

## **OPTIONS**

### **Global Options**

These options can be specified on the command line at any point before the first effect name.

#### **-h, --help**

Show version number and usage information.

#### **--help-effect=NAME**

Show usage information on the specified effect. The name **all** can be used to show usage on all effects.

#### **--help-format=NAME**

Show information about the specified file format. The name **all** can be used to show information on all formats.

#### **--buffer BYTES, --input-buffer BYTES**

Set the size in bytes of the buffers used for processing audio (default 8192). **--buffer** applies to input, effects, and output processing; **--input-buffer** applies only to input processing (for which it overrides **--buffer** if both are given).

Be aware that large values for **--buffer** will cause SoX to become slow to respond to requests to terminate or to skip the current input file.

**---effects-file=FILENAME**

Use FILENAME to obtain all effects and their arguments. The file is parsed as if the values were specified on the command line. A new line can be used in place of the special ":" marker to separate effect chains. This option causes any effects specified on the command line to be discarded.

**---interactive**

Prompt before overwriting an existing file with the same name as that given for the output file.

**N.B.** Unintentionally overwriting a file is easier than you might think, for example, if you accidentally enter

```
sox file1 file2 effect1 effect2 ...
```

when what you really meant was

```
play file1 file2 effect1 effect2 ...
```

then, without this option, file2 will be overwritten. Hence, using this option is strongly recommended; a 'shell' alias, script, or batch file may be an appropriate way of permanently enabling it.

**-m|-M|---combine concatenate|merge|mix|mix-power|sequence**

Select the input file combining method; **-m** selects 'mix', **-M** selects 'merge'.

See **Input File Combining** above for a description of the different combining methods.

**---plot gnuplot|octave|off**

If not set to **off** (the default if **---plot** is not given), run in a mode that can be used, in conjunction with the gnuplot program or the GNU Octave program, to assist with the selection and configuration of many of the transfer-function based effects. For the first given effect that supports the selected plotting program, SoX will output commands to plot the effect's transfer function, and then exit without actually processing any audio. E.g.

```
sox --plot octave input-file -n highpass 1320 > plot.m
octave plot.m
```

**-q, ---no-show-progress**

Run in quiet mode when SoX wouldn't otherwise do so; this is the opposite of the **-S** option.

**---replay-gain track|album|off**

Select whether or not to apply replay-gain adjustment to input files. The default is **off** for **sox** and **rec**, **album** for **play** where (at least) the first two input files are tagged with the same Artist and Album names, and **track** for **play** otherwise.

**-S, ---show-progress**

Display input file format/header information, and processing progress as input file(s) percentage complete, elapsed time, and remaining time (if known; shown in brackets), and the number of samples written to the output file. Also shown is a VU meter, and an indication if clipping has occurred. The VU meter shows up to two channels and is calibrated for digital audio as follows:



<i>dB FSD</i>	<i>Display</i> <i>(right channel)</i>
>=	
-25	-
-23	=
-21	==
-19	===
-17	===-
-15	====
-13	====-
-11	=====
-9	=====
-7	=====
-5	=====
-3	=====
-1	=====!

'In the red'

A three-second peak-held value of headroom in dBs will be shown to the right of the meter if this is below 6dB.

This option is enabled by default when using SoX to play or record audio.

#### **--version**

Show SoX's version number and exit.

#### **-V[level]**

Set verbosity. SoX displays messages on the console (stderr) according to the following verbosity levels:

- 0 No messages are shown at all; use the exit status to determine if an error has occurred.
- 1 Only error messages are shown. These are generated if SoX cannot complete the requested commands.
- 2 Warning messages are also shown. These are generated if SoX can complete the requested commands, but not exactly according to the requested command parameters, or if clipping occurs.
- 3 Descriptions of SoX's processing phases are also shown. Useful for seeing exactly how SoX is processing your audio.

4 and above

Messages to help with debugging SoX are also shown.

By default, the verbosity level is set to 2; each occurrence of the **-V** option increases the verbosity level by 1. Alternatively, the verbosity level can be set to an absolute number by specifying it immediately after the **-V**; e.g. **-V0** sets it to 0.

### **Input File Options**

These options apply only to input files and may precede only input filenames on the command line.

#### **-v, --volume FACTOR**

Adjust volume by a factor of *FACTOR*. This is a linear (amplitude) adjustment, so a number less than 1 decreases the volume; greater than 1 increases it. If a negative number is given, then in addition to the volume adjustment, the audio signal will be inverted.

See also the **stat** effect for information on how to find the maximum volume of an audio file; this can be used to help select suitable values for this option.

See also **Input File Balancing** above.

## Input & Output File Format Options

These options apply to the input or output file whose name they immediately precede on the command line and are used mainly when working with headerless file formats or when specifying a format for the output file that is different to that of the input file.

### **-b BITS, --bits BITS**

The number of bits in each encoded sample. Not applicable to complex encodings, e.g. MP3, GSM. Not necessary with encodings that have a fixed number of bits, e.g. A/ $\mu$ -law, ADPCM.

### **-1/-2/-3/-4/-8**

The number of bytes in each encoded sample. Aliases for **-b 8/-b 16/-b 24/-b 32/-b 64** respectively.

### **-c CHANNELS, --channels CHANNELS**

The number of audio channels in the audio file; this can be any number greater than zero. To cause the output file to have a different number of channels than the input file, include this option with the output file options. If the input and output file have a different number of channels then the **mixer** effect must be used. If the **mixer** effect is not specified on the command line it will be invoked internally with default parameters.

Alternatively, some effects (e.g. **synth**, **remix**) determine what will be the number of output channels; in this case, neither this option nor the **mixer** effect is necessary.

### **-e ENCODING, --encoding ENCODING**

The audio encoding type.

#### **signed-integer**

PCM data stored as signed ('two's complement') integers. Commonly used with a 16 or 24-bit encoding size. A value of 0 represents minimum signal power.

#### **unsigned-integer**

PCM data stored as signed ('two's complement') integers. Commonly used with an 8-bit encoding size. A value of 0 represents maximum signal power.

#### **floating-point**

PCM data stored as IEEE 753 single precision (32-bit) or double precision (64-bit) floating-point ('real') numbers. A value of 0 represents minimum signal power.

**a-law** International telephony standard for logarithmic encoding to 8 bits per sample. It has a precision equivalent to roughly 13-bit PCM and is sometimes encoded with reversed bit-ordering (see the **-X** option).

#### **u-law, mu-law**

North American telephony standard for logarithmic encoding to 8 bits per sample. A.k.a  $\mu$ -law. It has a precision equivalent to roughly 14-bit PCM and is sometimes encoded with reversed bit-ordering (see the **-X** option).

#### **oki-adpcm**

OKI (a.k.a. VOX, Dialogic, or Intel) 4-bit ADPCM; it has a precision equivalent to roughly 12-bit PCM. ADPCM is a form of audio compression that has a good compromise between audio quality and encoding/decoding speed.

#### **ima-adpcm**

IMA (a.k.a. DVI) 4-bit ADPCM; it has a precision equivalent to roughly 13-bit PCM.

#### **ms-adpcm**

Microsoft 4-bit ADPCM; it has a precision equivalent to roughly 14-bit PCM.

#### **gsm-full-rate**

GSM is currently used for the vast majority of the world's digital wireless telephone calls. It utilises several audio formats with different bit-rates and associated speech quality. SoX has support for GSM's original 13kbps 'Full Rate' audio format. It is usually CPU intensive to work with GSM audio.

Encoding names can be abbreviated where this would not be ambiguous; e.g. 'unsigned-integer' can be given as 'un', but not 'u' (ambiguous with 'u-law'). For reasons of forward compatibility, using abbreviations in scripts is not recommended.

Note that explicitly specifying other encoding types (e.g. MP3, FLAC) is not necessary since they can be inferred from the file type or header.

**-s/-u/-f/-A/-U/-o/-i/-a/-g**

Aliases for specifying the encoding types **signed-integer/unsigned-integer/floating-point/mu-law/a-law/oki-adpcm/ima-adpcm/ms-adpcm/gsm-full-rate** respectively.

**-r, --rate RATE[k]**

Gives the sample rate in Hz (or kHz if appended with 'k') of the file. To cause the output file to have a different sample rate than the input file, include this option with the output file format options.

If the input and output files have different rates then a sample rate change effect must be run. Since SoX has multiple rate changing effects, the user can specify which to use as an effect. If no rate change effect is specified then the **rate** effect will be chosen by default.

**-t, --type file-type**

Gives the type of the audio file. This is useful when the file extension is non-standard or when the type can not be determined by looking at the header of the file.

The **-t** option can also be used to override the type implied by an input filename extension, but if overriding with a type that has a header, SoX will exit with an appropriate error message if such a header is not actually present.

See **soxformat(7)** for a list of supported file types.

**-L, --endian little**

**-B, --endian big**

**-x, --endian swap**

These options specify whether the byte-order of the audio data is, respectively, 'little endian', 'big endian', or the opposite to that of the system on which SoX is being used. Endianness applies only to data encoded as signed or unsigned integers of 16 or more bits. It is often necessary to specify one of these options for headerless files, and sometimes necessary for (otherwise) self-describing files. A given endian-setting option may be ignored for an input file whose header contains a specific endianness identifier, or for an output file that is actually an audio device.

**N.B.** Unlike normal format characteristics, the endianness (byte, nibble, & bit ordering) of the input file is not automatically used for the output file; so, for example, when the following is run on a little-endian system:

```
sox -B audio.s2 trimmed.s2 trim 2
```

trimmed.s2 will be created as little-endian;

```
sox -B audio.s2 -B trimmed.s2 trim 2
```

must be used to preserve big-endianness in the output file.

The **-V** option can be used to check the selected orderings.

**-N, --reverse-nibbles**

Specifies that the nibble ordering (i.e. the 2 halves of a byte) of the samples should be reversed; sometimes useful with ADPCM-based formats.

**N.B.** See also N.B. in section on **-x** above.

**-X, --reverse-bits**

Specifies that the bit ordering of the samples should be reversed; sometimes useful with a few (mostly headerless) formats.

**N.B.** See also N.B. in section on **-x** above.

### Output File Format Options

These options apply only to the output file and may precede only the output filename on the command line.

**--add-comment** *TEXT*

Append a comment in the output file header (where applicable).

**--comment** *TEXT*

Specify the comment text to store in the output file header (where applicable).

SoX will provide a default comment if this option (or **--comment-file**) is not given; to specify that no comment should be stored in the output file, use **--comment ""**.

**--comment-file** *FILENAME*

Specify a file containing the comment text to store in the output file header (where applicable).

**-C, --compression** *FACTOR*

The compression factor for variably compressing output file formats. If this option is not given, then a default compression factor will apply. The compression factor is interpreted differently for different compressing file formats. See the description of the file formats that use this option in **soxformat(7)** for more information.

## EFFECTS

In addition to converting and playing audio files, SoX can be used to invoke a number of audio ‘effects’. Multiple effects may be applied by specifying them one after another at the end of the SoX command line; forming an effects chain. Note that applying multiple effects in real-time (i.e. when playing audio) is likely to need a high performance computer; stopping other applications may alleviate performance issues should they occur.

Some of the SoX effects are primarily intended to be applied to a single instrument or ‘voice’. To facilitate this, the **remix** effect and the global SoX option **-M** can be used to isolate then recombine tracks from a multi-track recording.

### Multiple Effect Chains

A single effects chain is made up of one or more effects. Audio from the input is run through the chain until either the input file reaches end of file or an effects in the chain requests to terminate the chain.

SoX supports running multiple effects chain over the input audio. In this case, when one chain indicates it is done processing audio the audio data is then sent through the next effects chain. This continues until either no more effects chains exist or the input has reached end of file.

An effects chain is terminated by placing a **:** (colon) after an effect. Any following effects are apart of a new effects chain.

It is important to place the effect that will stop the chain as the first effect in the chain. This is because any samples that are buffered by effects to the left of the terminating effect will be discarded. The amount of samples discarded is related to the **--buffer** option and it should be kept small, relative to the sample rate, if the terminating effect can not be first. Further information on stopping effects can be found in the **Stopping SoX** section.

There are a few pseudo-effects that aid using multiple effects chains. These include **newfile** which will start writing to a new output file before moving to the next effects chain and **restart** which will move back to the first effects chain. Pseudo-effects must be specified as the first effect in a chain and as the only effect in a chain (they must have a **:** before and after they are specified).

The following is an example of multiple effects chains. It will split the input file into multiple files of 30 seconds in length. Each output filename will have a unique number in its name as documented in **Output Files** section.

```
sox infile.wav output.wav trim 0 30 : newfile : restart
```

## Common Notation And Parameters

In the descriptions that follow, brackets [ ] are used to denote parameters that are optional, braces { } to denote those that are both optional and repeatable, and angle brackets < > to denote those that are repeatable but not optional. Where applicable, default values for optional parameters are shown in parenthesis ( ).

The following parameters are used with, and have the same meaning for, several effects:

*centre*[**k**]

See *frequency*.

*frequency*[**k**]

A frequency in Hz, or, if appended with 'k', kHz.

*gain*

A power gain in dB. Zero gives no gain; less than zero gives an attenuation.

*width*[**h**|**k**|**o**|**q**]

Used to specify the band-width of a filter. A number of different methods to specify the width are available (though not all for every effect); one of the characters shown may be appended to select the desired method as follows:

	<i>Method</i>	<i>Notes</i>
<b>h</b>	Hz	
<b>k</b>	kHz	
<b>o</b>	Octaves	
<b>q</b>	Q-factor	See [2]

For each effect that uses this parameter, the default method (i.e. if no character is appended) is the one that it listed first in the effect's first line of description.

To see if SoX has support for an optional effect, enter **sox -h** and look for its name under the list: 'EFFECTS'.

## Supported Effects

**allpass** *frequency*[**k**] *width*[**h**|**k**|**o**|**q**]

Apply a two-pole all-pass filter with central frequency (in Hz) *frequency*, and filter-width *width*. An all-pass filter changes the audio's frequency to phase relationship without changing its frequency to amplitude relationship. The filter is described in detail in [1].

This effect supports the **—plot** global option.

**band** [**-n**] *center*[**k**] [*width*[**h**|**k**|**o**|**q**]]

Apply a band-pass filter. The frequency response drops logarithmically around the *center* frequency. The *width* parameter gives the slope of the drop. The frequencies at *center* + *width* and *center* - *width* will be half of their original amplitudes. **band** defaults to a mode oriented to pitched audio, i.e. voice, singing, or instrumental music. The **-n** (for noise) option uses the alternate mode for un-pitched audio (e.g. percussion). **Warning:** **-n** introduces a power-gain of about 11dB in the filter, so beware of output clipping. **band** introduces noise in the shape of the filter, i.e. peaking at the *center* frequency and settling around it.

This effect supports the **—plot** global option.

See also **filter** for a bandpass filter with steeper shoulders.

**bandpass**|**bandreject** [**-c**] *frequency*[**k**] *width*[**h**|**k**|**o**|**q**]

Apply a two-pole Butterworth band-pass or band-reject filter with central frequency *frequency*, and (3dB-point) band-width *width*. The **-c** option applies only to **bandpass** and selects a constant skirt gain (peak gain = Q) instead of the default: constant 0dB peak gain. The filters roll off at 6dB per octave (20dB per decade) and are described in detail in [1].

These effects support the **—plot** global option.

See also **filter** for a bandpass filter with steeper shoulders.

**bandreject** *frequency*[k] *width*[h|k|o|q]

Apply a band-reject filter. See the description of the **bandpass** effect for details.

**bass|treble** *gain* [*frequency*[k] [*width*[s|h|k|o|q]]]

Boost or cut the bass (lower) or treble (upper) frequencies of the audio using a two-pole shelving filter with a response similar to that of a standard hi-fi's tone-controls. This is also known as shelving equalisation (EQ).

*gain* gives the gain at 0 Hz (for **bass**), or whichever is the lower of ~22 kHz and the Nyquist frequency (for **treble**). Its useful range is about -20 (for a large cut) to +20 (for a large boost). Beware of **Clipping** when using a positive *gain*.

If desired, the filter can be fine-tuned using the following optional parameters:

*frequency* sets the filter's central frequency and so can be used to extend or reduce the frequency range to be boosted or cut. The default value is 100 Hz (for **bass**) or 3 kHz (for **treble**).

*width* determines how steep is the filter's shelf transition. In addition to the common width specification methods described above, 'slope' (the default, or if appended with 's') may be used. The useful range of 'slope' is about 0.3, for a gentle slope, to 1 (the maximum), for a steep slope; the default value is 0.5.

The filters are described in detail in [1].

These effects support the **—plot** global option.

See also **equalizer** for a peaking equalisation effect.

**bend** [-f *frame-rate*(25)] [-o *over-sample*(16)] { *delay,cents,duration* }

Changes pitch by specified amounts at specified times. Each given triple: *delay,cents,duration* specifies one bend. *delay* is the amount of time after the start of the audio stream, or the end of the previous bend, at which to start bending the pitch; *cents* is the number of cents (100 cents = 1 semitone) by which to bend the pitch, and *duration* the length of time over which the pitch will be bent.

The pitch-bending algorithm utilises the Discrete Fourier Transform (DFT) at a particular frame rate and over-sampling rate. The -f and -o parameters may be used to adjust these parameters and thus control the smoothness of the changes in pitch.

For example, an initial tone is generated, then bent three times, yeilding four different notes in total:

```
play -n synth 2.5 sin 667 gain 1 \
    bend .35,180,.25 .15,740,.53 0,-520,.3
```

Note that the clipping that is produced in this example is deliberate; to remove it, use **gain -5** in place of **gain 1**.

**chorus** *gain-in gain-out* <*delay decay speed depth* -s|-t>

Add a chorus effect to the audio. This can make a single vocal sound like a chorus, but can also be applied to instrumentation.

Chorus resembles an echo effect with a short delay, but whereas with echo the delay is constant, with chorus, it is varied using sinusoidal or triangular modulation. The modulation depth defines the range the modulated delay is played before or after the delay. Hence the delayed sound will sound slower or faster, that is the delayed sound tuned around the original one, like in a chorus where some vocals are slightly off key. See [3] for more discussion of the chorus effect.

Each four-tuple parameter *delay/decay/speed/depth* gives the delay in milliseconds and the decay (relative to *gain-in*) with a modulation speed in Hz using *depth* in milliseconds. The modulation is either sinusoidal (-s) or triangular (-t). *Gain-out* is the volume of the output.

A typical delay is around 40ms to 60ms; the modulation speed is best near 0.25Hz and the

modulation depth around 2ms. For example, a single delay:

```
play guitar1.wav chorus 0.7 0.9 55 0.4 0.25 2 -t
```

Two delays of the original samples:

```
play guitar1.wav chorus 0.6 0.9 50 0.4 0.25 2 -t \
60 0.32 0.4 1.3 -s
```

A fuller sounding chorus (with three additional delays):

```
play guitar1.wav chorus 0.5 0.9 50 0.4 0.25 2 -t \
60 0.32 0.4 2.3 -t 40 0.3 0.3 1.3 -s
```

**compond** *attack1,decay1{,attack2,decay2}*  
*[soft-knee-dB:]in-dB1[,out-dB1]{,in-dB2,out-dB2}*  
*[gain [initial-volume-dB [delay]]]*

Compond (compress or expand) the dynamic range of the audio.

The *attack* and *decay* parameters (in seconds) determine the time over which the instantaneous level of the input signal is averaged to determine its volume; attacks refer to increases in volume and decays refer to decreases. For most situations, the attack time (response to the music getting louder) should be shorter than the decay time because the human ear is more sensitive to sudden loud music than sudden soft music. Where more than one pair of attack/decay parameters are specified, each input channel is companded separately and the number of pairs must agree with the number of input channels. Typical values are **0.3,0.8** seconds.

The second parameter is a list of points on the compander's transfer function specified in dB relative to the maximum possible signal amplitude. The input values must be in a strictly increasing order but the transfer function does not have to be monotonically rising. If omitted, the value of *out-dB1* defaults to the same value as *in-dB1*; levels below *in-dB1* are not companded (but may have gain applied to them). The point **0,0** is assumed but may be overridden (by **0,out-dBn**). If the list is preceded by a *soft-knee-dB* value, then the points at where adjacent line segments on the transfer function meet will be rounded by the amount given. Typical values for the transfer function are **6:-70,-60,-20**.

The third (optional) parameter is an additional gain in dB to be applied at all points on the transfer function and allows easy adjustment of the overall gain.

The fourth (optional) parameter is an initial level to be assumed for each channel when companding starts. This permits the user to supply a nominal level initially, so that, for example, a very large gain is not applied to initial signal levels before the companding action has begun to operate: it is quite probable that in such an event, the output would be severely clipped while the compander gain properly adjusts itself. A typical value (for audio which is initially quiet) is **-90** dB.

The fifth (optional) parameter is a delay in seconds. The input signal is analysed immediately to control the compander, but it is delayed before being fed to the volume adjuster. Specifying a delay approximately equal to the attack/decay times allows the compander to effectively operate in a 'predictive' rather than a reactive mode. A typical value is **0.2** seconds.

\* \* \*

The following example might be used to make a piece of music with both quiet and loud passages suitable for listening to in a noisy environment such as a moving vehicle:

```
sox asz.au asz-car.au compand 0.3,1 6:-70,-60,-20 -5 -90 0.2
```

The transfer function ('6:-70,...') says that very soft sounds (below -70dB) will remain unchanged. This will stop the compander from boosting the volume on 'silent' passages such as between movements. However, sounds in the range -60dB to 0dB (maximum volume) will be boosted so that the 60dB dynamic range of the original music will be compressed 3-to-1 into a

20dB range, which is wide enough to enjoy the music but narrow enough to get around the road noise. The '6:' selects 6dB soft-knee companding. The -5 (dB) output gain is needed to avoid clipping (the number is inexact, and was derived by experimentation). The -90 (dB) for the initial volume will work fine for a clip that starts with near silence, and the delay of 0.2 (seconds) has the effect of causing the compander to react a bit more quickly to sudden volume changes.

This effect supports the **--plot** global option (for the transfer function).

See also **mcompand** for a multiple-band companding effect.

#### **contrast** [*enhancement-amount* (75)]

Comparable with compression, this effect modifies an audio signal to make it sound louder. *enhancement-amount* controls the amount of the enhancement and is a number in the range 0–100. Note that *enhancement-amount* = 0 still gives a significant contrast enhancement. **contrast** is often used in conjunction with the **norm** effect as follows:

```
sox infile outfile norm -i contrast
```

#### **dcshift** *shift* [*limitergain*]

DC Shift the audio, with basic linear amplitude formula. This is most useful if your audio tends to not be centered around a value of 0. Shifting it back will allow you to get the most volume adjustments without clipping.

The first option is the *dcshift* value. It is a floating point number that indicates the amount to shift.

An optional *limitergain* can be specified as well. It should have a value much less than 1 (e.g. 0.05 or 0.02) and is used only on peaks to prevent clipping.

An alternative approach to removing a DC offset (albeit with a short delay) is to use the **highpass** filter effect at a frequency of say 10Hz, as illustrated in the following example:

```
sox -n out.au synth 5 sin %0 50 highpass 10
```

#### **deemph**

Apply ISO 908 de-emphasis (a treble attenuation shelving filter) to 44.1kHz (Compact Disc) audio.

Pre-emphasis was applied in the mastering of some CDs issued in the early 1980s. These included many classical music albums, as well as now sought-after issues of albums by The Beatles, Pink Floyd and others. Pre-emphasis should be removed at playback time by a de-emphasis filter in the playback device. However, not all modern CD players have this filter, and very few PC CD drives have it; playing pre-emphasised audio without the correct de-emphasis filter results in audio that sounds harsh and is far from what its creators intended.

With the **deemph** effect, it is possible to apply the necessary de-emphasis to audio that has been extracted from a pre-emphasised CD, and then either burn the de-emphasised audio to a new CD (which will then play correctly on any CD player), or simply play the correctly de-emphasised audio files on the PC. For example:

```
sox track1.wav track1-deemph.wav deemph
```

and then burn track1-deemph.wav to CD, or

```
play track1-deemph.wav
```

or simply

```
play track1.wav deemph
```

The de-emphasis filter is implemented as a biquad; its maximum deviation from the ideal response is only 0.06dB (up to 20kHz).

This effect supports the **--plot** global option.



See also the **bass** and **treble** shelving equalisation effects.

### **delay** {*length*}

Delay one or more audio channels. *length* can specify a time or, if appended with an 's', a number of samples. Do not specify both time and samples delays in the same command. For example, **delay 1.5 0 0.5** delays the first channel by 1.5 seconds, the third channel by 0.5 seconds, and leaves the second channel (and any other channels that may be present) un-delayed. The following (one long) command plays a chime sound:

```
play -n synth sin %-21.5 sin %-14.5 sin %-9.5 sin %-5.5 \
    sin %-2.5 sin %2.5 gain -5.4 fade h 0.008 2 1.5 \
    delay 0 .27 .54 .76 1.01 1.3 remix - fade h 0.1 2.72 2.5
```

### **dither** [-r|-t] [-s|-f *filter*] [*depth*]

Apply dithering to the audio. Dithering deliberately adds a small amount of noise to the signal in order to mask audible quantization effects that can occur if the output sample size is less than 24 bits. The default (or with the **-t** option) is Triangular (TPDF) white noise. The **-r** option can be used to select Rectangular Probability Density Function (RPDF) white noise. Noise-shaping (only for certain sample rates) can be selected with **-s**. With the **-f** option, it is possible to select a particular noise-shaping filter from the following list: lipshitz, f-weighted, modified-e-weighted, improved-e-weighted, gesemann, shibata, low-shibata, high-shibata. Note that most filter types are available only with 44100Hz sample rate. The filter types are distinguished by the following properties: audibility of noise, level of (inaudible, but in some circumstances, otherwise problematic) shaped high frequency noise, and processing speed.

By default, the amount of noise added is  $\pm\frac{1}{2}$  bit for RPDF,  $\pm 1$  bit for TPDF; the optional *depth* parameter (0.5 to 1) is a (linear or voltage) multiplier of this amount. Reducing this value reduces the audibility of the added white noise, but correspondingly creates residual quantization noise, so it should not normally be changed.

This effect should not be followed by any other effect that affects the audio.

### **earwax**

Makes audio easier to listen to on headphones. Adds 'cues' to 44.1kHz stereo (i.e. audio CD format) audio so that when listened to on headphones the stereo image is moved from inside your head (standard for headphones) to outside and in front of the listener (standard for speakers). See <http://www.geocities.com/beinges> for a full explanation.

### **echo** *gain-in gain-out* <*delay decay*>

Add echoing to the audio. Echoes are reflected sound and can occur naturally amongst mountains (and sometimes large buildings) when talking or shouting; digital echo effects emulate this behaviour and are often used to help fill out the sound of a single instrument or vocal. The time difference between the original signal and the reflection is the 'delay' (time), and the loudness of the reflected signal is the 'decay'. Multiple echoes can have different delays and decays.

Each given *delay decay* pair gives the delay in milliseconds and the decay (relative to gain-in) of that echo. Gain-out is the volume of the output. For example: This will make it sound as if there are twice as many instruments as are actually playing:

```
play lead.aiff echo 0.8 0.88 60 0.4
```

If the delay is very short, then it sound like a (metallic) robot playing music:

```
play lead.aiff echo 0.8 0.88 6 0.4
```

A longer delay will sound like an open air concert in the mountains:

```
play lead.aiff echo 0.8 0.9 1000 0.3
```

One mountain more, and:

```
play lead.aiff echo 0.8 0.9 1000 0.3 1800 0.25
```

**echos** *gain-in gain-out <delay decay>*

Add a sequence of echoes to the audio. Each *delay decay* pair gives the delay in milliseconds and the decay (relative to gain-in) of that echo. Gain-out is the volume of the output.

Like the echo effect, echos stand for ‘ECHO in Sequel’, that is the first echos takes the input, the second the input and the first echos, the third the input and the first and the second echos, ... and so on. Care should be taken using many echos; a single echos has the same effect as a single echo.

The sample will be bounced twice in symmetric echos:

```
play lead.aiff echos 0.8 0.7 700 0.25 700 0.3
```

The sample will be bounced twice in asymmetric echos:

```
play lead.aiff echos 0.8 0.7 700 0.25 900 0.3
```

The sample will sound as if played in a garage:

```
play lead.aiff echos 0.8 0.7 40 0.25 63 0.3
```

**equalizer** *frequency[k] width[q|o|h|k] gain*

Apply a two-pole peaking equalisation (EQ) filter. With this filter, the signal-level at and around a selected frequency can be increased or decreased, whilst (unlike band-pass and band-reject filters) that at all other frequencies is unchanged.

*frequency* gives the filter’s central frequency in Hz, *width*, the band-width, and *gain* the required gain or attenuation in dB. Beware of **Clipping** when using a positive *gain*.

In order to produce complex equalisation curves, this effect can be given several times, each with a different central frequency.

The filter is described in detail in [1].

This effect supports the **—plot** global option.

See also **bass** and **treble** for shelving equalisation effects.

**fade** *[type] fade-in-length [stop-time [fade-out-length]]*

Add a fade effect to the beginning, end, or both of the audio.

For fade-ins, this starts from the first sample and ramps the volume of the audio from 0 to full volume over *fade-in-length* seconds. Specify 0 seconds if no fade-in is wanted.

For fade-outs, the audio will be truncated at *stop-time* and the volume will be ramped from full volume down to 0 starting at *fade-out-length* seconds before the *stop-time*. If *fade-out-length* is not specified, it defaults to the same value as *fade-in-length*. No fade-out is performed if *stop-time* is not specified. If the file length can be determined from the input file header and length-changing effects are not in effect, then **0** may be specified for *stop-time* to indicate the usual case of a fade-out that ends at the end of the input audio stream.

All times can be specified in either periods of time or sample counts. To specify time periods use the format hh:mm:ss.frac format. To specify using sample counts, specify the number of samples and append the letter ‘s’ to the sample count (for example ‘8000s’).

An optional *type* can be specified to change the type of envelope. Choices are **q** for quarter of a sine wave, **h** for half a sine wave, **t** for linear slope, **l** for logarithmic, and **p** for inverted parabola. The default is logarithmic.

**filter** *[low]–[high] [window-len [beta]]*

Apply a sinc-windowed lowpass, highpass, or bandpass filter of given window length to the signal. *low* refers to the frequency of the lower 6dB corner of the filter. *high* refers to the frequency of the upper 6dB corner of the filter.

A low-pass filter is obtained by leaving *low* unspecified, or 0. A high-pass filter is obtained by

leaving *high* unspecified, or 0, or greater than or equal to the Nyquist frequency.

The *window-len*, if unspecified, defaults to 128. Longer windows give a sharper cut-off, smaller windows a more gradual cut-off.

The *beta* parameter determines the type of filter window used. Any value greater than 2 is the beta for a Kaiser window.  $\text{Beta} \leq 2$  selects a Blackman-Nuttall window. If unspecified, the default is a Kaiser window with beta 16.

In the case of Kaiser window ( $\text{beta} > 2$ ), lower betas produce a somewhat faster transition from pass-band to stop-band, at the cost of noticeable artifacts. A beta of 16 is the default, beta less than 10 is not recommended. If you want a sharper cut-off, don't use low beta's, use a longer sample window. A Blackman-Nuttall window is selected by specifying any 'beta'  $\leq 2$ , and the Blackman-Nuttall window has somewhat steeper cut-off than the default Kaiser window. You will probably not need to use the beta parameter at all, unless you are just curious about comparing the effects of Blackman-Nuttall vs. Kaiser windows.

This effect supports the **—plot** global option.

**flanger** [*delay depth regen width speed shape phase interp*]

Apply a flanging effect to the audio. See [3] for a detailed description of flanging.

All parameters are optional (right to left).

	Range	Default	Description
<i>delay</i>	0 – 10	0	Base delay in milliseconds.
<i>depth</i>	0 – 10	2	Added swept delay in milliseconds.
<i>regen</i>	–95 – 95	0	Percentage regeneration (delayed signal feedback).
<i>width</i>	0 – 100	71	Percentage of delayed signal mixed with original.
<i>speed</i>	0.1 – 10	0.5	Sweeps per second (Hz).
<i>shape</i>		sin	Swept wave shape: <b>sine triangle</b> .
<i>phase</i>	0 – 100	25	Swept wave percentage phase-shift for multi-channel (e.g. stereo) flange; 0 = 100 = same phase on each channel.
<i>interp</i>		lin	Digital delay-line interpolation: <b>linear quadratic</b> .

**gain** *dB-gain*

Apply an amplification or an attenuation to the audio signal. The signal level is adjusted by the given number of dB—positive amplifies (beware of Clipping), negative attenuates.

See also the **vol** effect.

**highpass|lowpass** [**–1|–2**] *frequency*[**k**] [*width*[**q|o|h|k**]]

Apply a high-pass or low-pass filter with 3dB point *frequency*. The filter can be either single-pole (with **–1**), or double-pole (the default, or with **–2**). *width* applies only to double-pole filters; the default is  $Q = 0.707$  and gives a Butterworth response. The filters roll off at 6dB per pole per octave (20dB per pole per decade). The double-pole filters are described in detail in [1].

These effects support the **—plot** global option.

See also **filter** for filters with a steeper roll-off.

**ladspa module** [**plugin**] [**argument...**]

Apply a LADSPA [5] (Linux Audio Developer's Simple Plugin API) plugin. Despite the name, LADSPA is not Linux-specific, and a wide range of effects is available as LADSPA plugins, such as cmt [6] (the Computer Music Toolkit) and Steve Harris's plugin collection [7]. The first argument is the plugin module, the second the name of the plugin (a module can contain more than one

plugin) and any other arguments are for the control ports of the plugin. Missing arguments are supplied by default values if possible. Only plugins with at most one audio input and one audio output port can be used. If found, the environment variable LADSPA\_PATH will be used as search path for plugins.

#### **loudness** [*gain* [*reference*]]

Loudness control—similar to the **gain** effect, but provides equalisation for the human auditory system. See <http://en.wikipedia.org/wiki/Loudness> for a detailed description of loudness. The gain is adjusted by the given *gain* parameter (usually negative) and the signal equalised according to ISO 226 w.r.t. a reference level of 65dB, though an alternative *reference* level may be given if the original audio has been equalised for some other optimal level. A default gain of -10dB is used if a *gain* value is not given.

See also the **gain** effect.

#### **lowpass** [-1|-2] *frequency*[**k**] [width[**q**|**o**|**h**|**k**]]

Apply a low-pass filter. See the description of the **highpass** effect for details.

#### **mcompand** "*attack1,decay1*{*attack2,decay2*}

[*soft-knee-dB*:]*in-dB1*[*out-dB1*]{*in-dB2*,*out-dB2*}

[*gain* [*initial-volume-dB* [*delay*]]]" {*crossover-freq*[**k**] "*attack1*,..."}

The multi-band compander is similar to the single-band compander but the audio is first divided into bands using Linkwitz-Riley cross-over filters and a separately specifiable compander run on each band. See the **compand** effect for the definition of its parameters. Compand parameters are specified between double quotes and the crossover frequency for that band is given by *crossover-freq*; these can be repeated to create multiple bands.

For example, the following (one long) command shows how multi-band companding is typically used in FM radio:

```
play track1.wav gain -3 filter 8000- 32 100 mcompand \
  "0.005,0.1 -47,-40,-34,-34,-17,-33" 100 \
  "0.003,0.05 -47,-40,-34,-34,-17,-33" 400 \
  "0.000625,0.0125 -47,-40,-34,-34,-15,-33" 1600 \
  "0.0001,0.025 -47,-40,-34,-34,-31,-31,-0,-30" 6400 \
  "0,0.025 -38,-31,-28,-28,-0,-25" \
  gain 15 highpass 22 highpass 22 filter -17500 256 \
  gain 9 lowpass -1 17801
```

The audio file is played with a simulated FM radio sound (or broadcast signal condition if the low-pass filter at the end is skipped). Note that the pipeline is set up with US-style 75us preemphasis.

See also **compand** for a single-band companding effect.

#### **mixer** [-l|-r|-f|-b|-1|-2|-3|-4|*n*{*n*} ]

Reduce the number of audio channels by mixing or selecting channels, or increase the number of channels by duplicating channels. Note: this effect operates on the audio *channels* within the SoX effects processing chain; it should not be confused with the **-m** global option (where multiple *files* are mix-combined before entering the effects chain).

This effect is automatically used when the number of input channels differ from the number of output channels. When reducing the number of channels it is possible to manually specify the **mixer** effect and use the **-l**, **-r**, **-f**, **-b**, **-1**, **-2**, **-3**, **-4**, options to select only the left, right, front, back channel(s) or specific channel for the output instead of averaging the channels. The **-l**, and **-r** options will do averaging in quad-channel files so select the exact channel to prevent this.

The **mixer** effect can also be invoked with up to 16 numbers, separated by commas, which specify the proportion (0 = 0% and 1 = 100%) of each input channel that is to be mixed into each output channel. In two-channel mode, 4 numbers are given:  $l \rightarrow l$ ,  $l \rightarrow r$ ,  $r \rightarrow l$ , and  $r \rightarrow r$ , respectively. In four-channel mode, the first 4 numbers give the proportions for the left-front output channel, as

follows:  $lf \rightarrow lf$ ,  $rf \rightarrow lf$ ,  $lb \rightarrow lf$ , and  $rb \rightarrow rf$ . The next 4 give the right-front output in the same order, then left-back and right-back.

It is also possible to use the 16 numbers to expand or reduce the channel count; just specify 0 for unused channels.

Finally, certain reduced combination of numbers can be specified for certain input/output channel combinations.

In Ch	Out Ch	Num	Mappings
2	1	2	$l \rightarrow l, r \rightarrow l$
2	2	1	adjust balance
4	1	4	$lf \rightarrow l, rf \rightarrow l, lb \rightarrow l, rb \rightarrow l$
4	2	2	$lf \rightarrow l\&rf \rightarrow r, lb \rightarrow l\&rb \rightarrow r$
4	4	1	adjust balance
4	4	2	front balance, back balance

See also **remix** for a mixing effect that handles any number of channels.

#### **noiseprof** [*profile-file*]

Calculate a profile of the audio for use in noise reduction. See the description of the **noisered** effect for details.

#### **noisered** [*profile-file* [*amount*]]

Reduce noise in the audio signal by profiling and filtering. This effect is moderately effective at removing consistent background noise such as hiss or hum. To use it, first run SoX with the **noiseprof** effect on a section of audio that ideally would contain silence but in fact contains noise—such sections are typically found at the beginning or the end of a recording. **noiseprof** will write out a noise profile to *profile-file*, or to stdout if no *profile-file* or if ‘-’ is given. E.g.

```
sox speech.au -n trim 0 1.5 noiseprof speech.noise-profile
```

To actually remove the noise, run SoX again, this time with the **noisered** effect; **noisered** will reduce noise according to a noise profile (which was generated by **noiseprof**), from *profile-file*, or from stdin if no *profile-file* or if ‘-’ is given. E.g.

```
sox speech.au cleaned.au noisered speech.noise-profile 0.3
```

How much noise should be removed is specified by *amount*—a number between 0 and 1 with a default of 0.5. Higher numbers will remove more noise but present a greater likelihood of removing wanted components of the audio signal. Before replacing an original recording with a noise-reduced version, experiment with different *amount* values to find the optimal one for your audio; use headphones to check that you are happy with the results, paying particular attention to quieter sections of the audio.

On most systems, the two stages—profiling and reduction—can be combined using a pipe, e.g.

```
sox noisy.au -n trim 0 1 noiseprof | play noisy.au noisered
```

#### **norm** [-i|-b] [*level*]

Normalise audio to 0dB FSD, to a given level relative to 0dB, or normalise the balance of multi-channel audio. Requires temporary file space to store the audio to be normalised.

To create a normalised copy of an audio file,

```
sox infile outfile norm
```

can be used, though note that if ‘infile’ has a simple encoding (e.g. PCM), then

```
sox infile outfile vol `sox infile -n stat -v 2>&1`
```

(on systems that support this construct) might be quicker to execute (though perhaps not to type!) as it doesn’t require a temporary file.

For a more complex example, suppose that ‘effect1’ performs some unknown or unpredictable attenuation and that ‘effect2’ requires up to 10dB of headroom, then

```
sox infile outfile effect1 norm -10 effect2 norm
```

gives both effect2 and the output file the highest possible signal levels.

Normally, audio is normalised based on the level of the channel with the highest peak level, which means that whilst all channels are adjusted, only one channel attains the normalised level. If the **-i** option is given, then each channel is treated individually and will attain the normalised level.

If the **-b** option is given (with a multi-channel audio file), then the audio channels will be balanced; i.e. the RMS level of each channel will be normalised to that of the channel with the highest RMS level. This can be used, for example, to correct stereo imbalance. Note that **-b** can cause clipping.

In most cases, **norm -3** should be the maximum level used at the output file (to leave headroom for playback-resampling, etc.). See also the discussions of **Clipping** and **Replay Gain** above.

**oops** Out Of Phase Stereo effect. Mixes stereo to twin-mono where each mono channel contains the difference between the left and right stereo channels. This is sometimes known as the ‘karaoke’ effect as it often has the effect of removing most or all of the vocals from a recording.

**pad** { *length*[@*position*] }

Pad the audio with silence, at the beginning, the end, or any specified points through the audio. Both *length* and *position* can specify a time or, if appended with an ‘s’, a number of samples. *length* is the amount of silence to insert and *position* the position in the input audio stream at which to insert it. Any number of lengths and positions may be specified, provided that a specified position is not less than the previous one. *position* is optional for the first and last lengths specified and if omitted correspond to the beginning and the end of the audio respectively. For example, **pad 1.5 1.5** adds 1.5 seconds of silence padding at each end of the audio, whilst **pad 4000s@3:00** inserts 4000 samples of silence 3 minutes into the audio. If silence is wanted only at the end of the audio, specify either the end position or specify a zero-length pad at the start.

**phaser** *gain-in gain-out delay decay speed* [-s|-t]

Add a phasing effect to the audio. See [3] for a detailed description of phasing.

*delay/decay/speed* gives the delay in milliseconds and the decay (relative to gain-in) with a modulation speed in Hz. The modulation is either sinusoidal (-s) —preferable for multiple instruments, or triangular (-t) —gives single instruments a sharper phasing effect. The decay should be less than 0.5 to avoid feedback, and usually no less than 0.1. Gain-out is the volume of the output.

For example:

```
play snare.flac phaser 0.8 0.74 3 0.4 0.5 -t
```

Gentler:

```
play snare.flac phaser 0.9 0.85 4 0.23 1.3 -s
```

A popular sound:

```
play snare.flac phaser 0.89 0.85 1 0.24 2 -t
```

More severe:

```
play snare.flac phaser 0.6 0.66 3 0.6 2 -t
```

**pitch** [-q] *shift* [*segment* [*search* [*overlap*]]]

Change the audio pitch (but not tempo).

*shift* gives the pitch shift as positive or negative ‘cents’ (i.e. 100ths of a semitone). See the **tempo** effect for a description of the other parameters.

**rate** [-q|-l|-m|-h|-v] [override-options] *RATE*[k]

Change the audio sampling rate (i.e. resample the audio) to any given *RATE* (even non-integer if this is supported by the output file format) using a quality level defined as follows:

	<i>Quality</i>	<i>Band-width</i>	<i>Rej dB</i>	<i>Typical Use</i>
-q	quick	n/a	≈30 @ Fs/4	playback on ancient hardware
-l	low	80%	100	playback on old hardware
-m	medium	95%	100	audio playback
-h	high	95%	125	16-bit mastering (use with dither)
-v	very high	95%	175	24-bit mastering

where *Band-width* is the percentage of the audio frequency band that is preserved and *Rej dB* is the level of noise rejection. Increasing levels of resampling quality come at the expense of increasing amounts of time to process the audio. If no quality option is given, the quality level used is 'high'.

The 'quick' algorithm uses cubic interpolation; all others use band-limited interpolation. By default, all algorithms have a 'linear' phase response; for 'medium', 'high' and 'very high', the phase response is configurable (see below).

The **rate** effect is invoked automatically if SoX's -r option specifies a rate that is different to that of the input file(s). Alternatively, if this effect is given explicitly, then SoX's -r option need not be given. For example, the following two commands are equivalent:

```
sox input.au -r 48k output.au bass -3
sox input.au          output.au bass -3 rate 48k
```

though the second command is more flexible as it allows **rate** options to be given, and allows the effects to be ordered arbitrarily.

\* \* \*

Warning: technically detailed discussion follows.

The simple quality selection described above provides settings that satisfy the needs of the vast majority of resampling tasks. Occasionally, however, it may be desirable to fine-tune the resampler's filter response; this can be achieved using *override options*, as detailed in the following table:

-M/-I/-L	Phase response = minimum/intermediate/linear
-s	Steep filter (band-width = 99%)
-a	Allow aliasing above the pass-band
-b 74-99.7	Any band-width %
-p 0-100	Any phase response (0 = minimum, 25 = intermediate, 50 = linear, 100 = maximum)

N.B. Override options can not be used with the 'quick' or 'low' quality algorithms.

All resamplers use filters that can sometimes create 'echo' (a.k.a. 'ringing') artefacts with transient signals such as those that occur with 'finger snaps' or other highly percussive sounds. Such artefacts are much more noticable to the human ear if they occur before the transient ('pre-echo') than if they occur after it ('post-echo'). Note that frequency of any such artefacts is related to the smaller of the original and new sampling rates but that if this is at least 44.1kHz, then the artefacts will lie outside the range of human hearing.

A phase response setting may be used to control the distribution of any transient echo between 'pre' and 'post': with minimum phase, there is no pre-echo but the longest post-echo; with linear

phase, pre and post echo are in equal amounts (in signal terms, but not audibility terms); the intermediate phase setting attempts to find the best compromise by selecting a small length (and level) of pre-echo and a medium lengthed post-echo.

Minimum, intermediate, or linear phase response is selected using the **-M**, **-I**, or **-L** option; a custom phase response can be created with the **-p** option. Note that phase responses between ‘linear’ and ‘maximum’ (greater than 50) are rarely useful.

A resampler’s band-width setting determines how much of the frequency content of the original signal (w.r.t. the original sample rate when up-sampling, or the new sample rate when down-sampling) is preserved during conversion. The term ‘pass-band’ is used to refer to all frequencies up to the band-width point (e.g. for 44.1kHz sampling rate, and a resampling band-width of 95%, the pass-band represents frequencies from 0Hz (D.C.) to circa 21kHz). Increasing the resampler’s band-width results in a slower conversion and can increase transient echo artefacts (and vice versa).

The **-s** ‘steep filter’ option changes resampling band-width from the default 95% (based on the 3dB point), to 99%. The **-b** option allows the band-width to be set to any value in the range 74–99.7 %, but note that band-width values greater than 99% are not recommended for normal use as they can cause excessive transient echo.

If the **-a** option is given, then aliasing above the pass-band is allowed. For example, with 44.1kHz sampling rate, and a resampling band-width of 95%, this means that frequency content above 21kHz can be distorted; however, since this is above the pass-band (i.e. above the highest frequency of interest/audibility), this may not be a problem. The benefits of allowing aliasing are reduced processing time, and reduced (by almost half) transient echo artefacts. Note that if this option is given, then the minimum band-width allowable with **-b** increases to 85%.

Examples:

```
sox input.wav -b 16 output.wav rate -s -a 44100 dither
```

default (high) quality resampling; overrides: steep filter, allow aliasing; to 44.1kHz sample rate; dither output to 16-bit WAV file.

```
sox input.wav -b 24 output.aiff rate -v -L -b 90 48k
```

very high quality resampling; overrides: linear phase, band-width 90%; to 48k sample rate; store output to 24-bit AIFF file.

\* \* \*

The **pitch**, **speed** and **tempo** effects all use the **rate** effect at their core.

See also **resample**, **polyphase** and **rabbit** for other sample-rate changing effects.

**remix** [**-a**|-**m**|-**p**] <out-spec>  
*out-spec* = *in-spec*{,*in-spec*} | **0**  
*in-spec* = [*in-chan*][-*in-chan2*][*vol-spec*]  
*vol-spec* = **p**|**i**|**v**[*volume*]

Select and mix input audio channels into output audio channels. Each output channel is specified, in turn, by a given *out-spec*: a list of contributing input channels and volume specifications.

Note that this effect operates on the audio *channels* within the SoX effects processing chain; it should not be confused with the **-m** global option (where multiple *files* are mix-combined before entering the effects chain).

An *out-spec* contains comma-separated input channel-numbers and hyphen-delimited channel-number ranges; alternatively, **0** may be given to create a silent output channel. For example,

```
sox input.au output.au remix 6 7 8 0
```

creates an output file with four channels, where channels 1, 2, and 3 are copies of channels 6, 7,



and 8 in the input file, and channel 4 is silent. Whereas

```
sox input.au output.au remix 1-3,7 3
```

creates a (somewhat bizarre) stereo output file where the left channel is a mix-down of input channels 1, 2, 3, and 7, and the right channel is a copy of input channel 3.

Where a range of channels is specified, the channel numbers to the left and right of the hyphen are optional and default to 1 and to the number of input channels respectively. Thus

```
sox input.au output.au remix -
```

performs a mix-down of all input channels to mono.

By default, where an output channel is mixed from multiple ( $n$ ) input channels, each input channel will be scaled by a factor of  $1/n$ . Custom mixing volumes can be set by following a given input channel or range of input channels with a *vol-spec* (volume specification). This is one of the letters **p**, **i**, or **v**, followed by a volume number, the meaning of which depends on the given letter and is defined as follows:

Letter	Volume number	Notes
p	power adjust in dB	0 = no change
i	power adjust in dB	As 'p', but invert the audio
v	voltage multiplier	1 = no change, 0.5 $\approx$ 6dB attenuation, 2 $\approx$ 6dB gain, -1 = invert

If an *out-spec* includes at least one *vol-spec* then, by default,  $1/n$  scaling is not applied to any other channels in the same out-spec (though may be in other out-specs). The **-a** (automatic) option however, can be given to retain the automatic scaling in this case. For example,

```
sox input.au output.au remix 1,2 3,4v0.8
```

results in channel level multipliers of 0.5,0.5 1,0.8, whereas

```
sox input.au output.au remix -a 1,2 3,4v0.8
```

results in channel level multipliers of 0.5,0.5 0.5,0.8.

The **-m** (manual) option disables all automatic volume adjustments, so

```
sox input.au output.au remix -m 1,2 3,4v0.8
```

results in channel level multipliers of 1,1 1,0.8.

The volume number is optional and omitting it corresponds to no volume change; however, the only case in which this is useful is in conjunction with **i**. For example, if *input.au* is stereo, then

```
sox input.au output.au remix 1,2i
```

is a mono equivalent of the **oops** effect.

If the **-p** option is given, then any automatic  $1/n$  scaling is replaced by  $1/\sqrt{n}$  ('power') scaling; this gives a louder mix but one that might occasionally clip.

\* \* \*

One use of the **remix** effect is to split an audio file into a set of files, each containing one of the constituent channels (in order to perform subsequent processing on individual audio channels). Where more than a few channels are involved, a script such as the following (Bourne shell script) is useful:

```
#!/bin/sh
chans=`soxi -c "$1"`
while [ $chans -ge 1 ]; do
    chans0=`printf %02i $chans`    # 2 digits hence up to 99 chans
```

```

out=`echo "$1" | sed "s/\(.*\)\\. \(.*\)\/\1-$chans0.\2/"`
sox "$1" "$out" remix $chans
chans=`expr $chans - 1`
done

```

If a file *input.au* containing six audio channels were given, the script would produce six output files: *input-01.au*, *input-02.au*, ..., *input-06.au*.

See also **mixer** and **swap** for similar effects.

#### **repeat** *count*

Repeat the entire audio *count* times. Requires temporary file space to store the audio to be repeated. Note that repeating once yields two copies: the original audio and the repeated audio.

**reverb** [**-w**|**--wet-only**] [*reverberance* (50%)] [*HF-damping* (50%)]  
 [*room-scale* (100%)] [*stereo-depth* (100%)]  
 [*pre-delay* (0ms)] [*wet-gain* (0dB)]]]]]]

Add reverberation to the audio using the ‘freeverb’ algorithm. A reverberation effect is sometimes desirable for concert halls that are too small or contain so many people that the hall’s natural reverberance is diminished. Applying a small amount of stereo reverb to a (dry) mono signal will usually make it sound more natural. See [3] for a detailed description of reverberation.

Note that this effect increases both the volume and the length of the audio, so to prevent clipping in these domains, a typical invocation might be:

```
play dry.au gain -3 pad 0 3 reverb
```

**reverse** Reverse the audio completely. Requires temporary file space to store the audio to be reversed.

**riaa** Apply RIAA vinyl playback equalisation. The sampling rate must be one of: 44.1, 48, 88.2, 96 kHz.

This effect supports the **--plot** global option.

**silence** [**-l**] [*above-periods* [*duration* threshold[**d** | %]] [*below-periods* *duration* threshold[**d** | %]]]

Removes silence from the beginning, middle, or end of the audio. Silence is anything below a specified threshold.

The *above-periods* value is used to indicate if audio should be trimmed at the beginning of the audio. A value of zero indicates no silence should be trimmed from the beginning. When specifying an non-zero *above-periods*, it trims audio up until it finds non-silence. Normally, when trimming silence from beginning of audio the *above-periods* will be 1 but it can be increased to higher values to trim all audio up to a specific count of non-silence periods. For example, if you had an audio file with two songs that each contained 2 seconds of silence before the song, you could specify an *above-period* of 2 to strip out both silence periods and the first song.

When *above-periods* is non-zero, you must also specify a *duration* and *threshold*. *Duration* indicates the amount of time that non-silence must be detected before it stops trimming audio. By increasing the duration, burst of noise can be treated as silence and trimmed off.

*Threshold* is used to indicate what sample value you should treat as silence. For digital audio, a value of 0 may be fine but for audio recorded from analog, you may wish to increase the value to account for background noise.

When optionally trimming silence from the end of the audio, you specify a *below-periods* count. In this case, *below-period* means to remove all audio after silence is detected. Normally, this will be a value 1 of but it can be increased to skip over periods of silence that are wanted. For example, if you have a song with 2 seconds of silence in the middle and 2 second at the end, you could set below-period to a value of 2 to skip over the silence in the middle of the audio.

For *below-periods*, *duration* specifies a period of silence that must exist before audio is not copied any more. By specifying a higher duration, silence that is wanted can be left in the audio. For example, if you have a song with an expected 1 second of silence in the middle and 2 seconds of silence at the end, a duration of 2 seconds could be used to skip over the middle silence.

Unfortunately, you must know the length of the silence at the end of your audio file to trim off silence reliably. A work around is to use the **silence** effect in combination with the **reverse** effect. By first reversing the audio, you can use the *above-periods* to reliably trim all audio from what looks like the front of the file. Then reverse the file again to get back to normal.

To remove silence from the middle of a file, specify a *below-periods* that is negative. This value is then treated as a positive value and is also used to indicate the effect should restart processing as specified by the *above-periods*, making it suitable for removing periods of silence in the middle of the audio.

The option **-l** indicates that *below-periods duration* length of audio should be left intact at the beginning of each period of silence. For example, if you want to remove long pauses between words but do not want to remove the pauses completely.

The *period* counts are in units of samples. *Duration* counts may be in the format of hh:mm:ss.frac, or the exact count of samples. *Threshold* numbers may be suffixed with **d** to indicate the value is in decibels, or **%** to indicate a percentage of maximum value of the sample value (**0%** specifies pure digital silence).

The following example shows how this effect can be used to start a recording that does not contain the delay at the start which usually occurs between ‘pressing the record button’ and the start of the performance:

```
rec parameters filename other-effects silence 1 5 2%
```

### **speed** *factor*[c]

Adjust the audio speed (pitch and tempo together). *factor* is either the ratio of the new speed to the old speed: greater than 1 speeds up, less than 1 slows down, or, if appended with the letter ‘c’, the number of cents (i.e. 100ths of a semitone) by which the pitch (and tempo) should be adjusted: greater than 0 increases, less than 0 decreases.

By default, the speed change is performed by resampling with the **rate** effect using its default quality/speed. For higher quality or higher speed resampling, in addition to the **speed** effect, specify the **rate** effect with the desired quality option.

### **spectrogram** [options]

Create a spectrogram of the audio. This effect is optional; type **sox --help** and check the list of supported effects to see if it has been included.

The spectrogram is rendered in a Portable Network Graphic (PNG) file, and shows time in the X-axis, frequency in the Y-axis, and audio signal magnitude in the Z-axis. Z-axis values are represented by the colour (or intensity) of the pixels in the X-Y plane.

This effect supports only one channel; for multi-channel input files, use either SoX’s **-c 1** option with the output file (to obtain a spectrogram on the mix-down), or the **remix n** effect to select a particular channel. Be aware though, that both of these methods affect the audio in the effects chain.

**-x num** X-axis pixels/second, default 100. This controls the width of the spectrogram; *num* can be from 1 (low time resolution) to 5000 (high time resolution) and need not be an integer. SoX may make a slight adjustment to the given number for processing quantisation reasons; if so, SoX will report the actual number used (viewable when **--verbose** is in effect).

The maximum width of the spectrogram is 999 pixels; if the audio length and the given **-x** number are such that this would be exceeded, then the spectrogram (and the effects

chain) will be truncated. To move the spectrogram to a point later in the audio stream, first invoke the **trim** effect; e.g.

```
sox audio.ogg -n trim 1:00 spectrogram
```

starts the spectrogram at 1 minute through the audio.

- y num** Y-axis resolution (1 – 4), default 2. This controls the height of the spectrogram; *num* can be from 1 (low frequency resolution) to 4 (high frequency resolution). For values greater than 2, the resulting image may be too tall to display on the screen; if so, a graphic manipulation package (such as **ImageMagick**(1)) can be used to re-size the image.

To increase the frequency resolution without increasing the height of the spectrogram, the **rate** effect may be invoked to reduce the sampling rate of the signal before invoking **spectrogram**; e.g.

```
sox audio.ogg -r 4k -n rate spectrogram
```

allows detailed analysis of frequencies up to 2kHz (half the sampling rate).

- z num** Z-axis (colour) range in dB, default 120. This sets the dynamic-range of the spectrogram to be  $-num$  dBFS to 0 dBFS. *Num* may range from 20 to 180. Decreasing dynamic-range effectively increases the ‘contrast’ of the spectrogram display, and vice versa.

**-Z num**

Sets the upper limit of the Z-axis in dBFS. A negative *num* effectively increases the ‘brightness’ of the spectrogram display, and vice versa.

- q num** Sets the Z-axis quantisation, i.e. the number of different colours (or intensities) in which to render Z-axis values. A small number (e.g. 4) will give a ‘poster’-like effect making it easier to discern magnitude bands of similar level. Small numbers also usually result in small PNG files. The number given specifies the number of colours to use inside the Z-axis range; two colours are reserved to represent out-of-range values.

**-w name**

Window: Hann (default), Hamming, Bartlett, Rectangular or Kaiser. The spectrogram is produced using the Discrete Fourier Transform (DFT) algorithm. A significant parameter to this algorithm is the choice of ‘window function’. By default, SoX uses the Hann window which has good all-round frequency-resolution and dynamic-range properties. For better frequency resolution (but lower dynamic-range), select a Hamming window; for higher dynamic-range (but poorer frequency-resolution), select a Kaiser window. Bartlett and Rectangular windows are also available. Selecting a window other than Hann will usually require a corresponding **-z** setting.

- s** Allow slack overlapping of DFT windows. This can, in some cases, increase image sharpness and give greater adherence to the **-x** value, but at the expense of a little spectral loss.

- m** Creates a monochrome spectrogram (the default is colour).

- h** Selects a high-colour palette—less visually pleasing than the default colour palette, but it may make it easier to differentiate different levels. If this option is used in conjunction with **-m**, the result will be a hybrid monochrome/colour palette.

- p num** Permute the colours in a colour or hybrid palette. The *num* parameter (from 1 to 6) selects the permutation.

- l** Creates a ‘printer friendly’ spectrogram with a light background (the default has a dark background).

- a** Suppress the display of the axis lines. This is sometimes useful in helping to discern artefacts at the spectrogram edges.

- t text** Set the image title—text to display above the spectrogram.
- c text** Set the image comment—text to display below and to the left of the spectrogram.
- o text** Name of the spectrogram output PNG file, default 'spectrogram.png'.

For example, let's see what the spectrogram of a swept triangular wave looks like:

```
sox -n -n synth 6 tri 10k:14k spectrogram -z 100 -w k
```

Append the following to the 'chime' example in the **delay** effect to see its spectrogram:

```
rate 2k spectrogram -x 200 -Z -15 -w k
```

For the ability to perform off-line processing of spectral data, see the **stat** effect.

**splice** { *position*[,*excess*[,*leeway*]] }

Splice together audio sections. This effect provides two things over simple audio concatenation: a (usually short) cross-fade is applied at the join, and a wave similarity comparison is made to help determine the best place at which to make the join.

To perform a splice, first use the **trim** effect to select the audio sections to be joined together. As when performing a tape splice, the end of the section to be spliced onto should be trimmed with a small *excess* (default 0.005 seconds) of audio after the ideal joining point. The beginning of the audio section to splice on should be trimmed with the same *excess* (before the ideal joining point), plus an additional *leeway* (default 0.005 seconds). SoX should then be invoked with the two audio sections as input files and the **splice** effect given with the position at which to perform the splice—this is length of the first audio section (including the excess).

For example, a long song begins with two verses which start (as determined e.g. by using the **play** command with the **trim** (*start*) effect) at times 0:30.125 and 1:03.432. The following commands cut out the first verse:

```
sox too-long.au part1.au trim 0 30.130
```

(5 ms excess, after the first verse starts)

```
sox long.au part2.au trim 1:03.422
```

(5 ms excess plus 5 ms leeway, before the second verse starts)

```
sox part1.au part2.au just-right.au splice 30.130
```

Provided your arithmetic is good enough, multiple splices can be performed with a single **splice** invocation. For example:

```
#!/bin/sh
# Audio Copy and Paste Over
# acpo infile copy-start copy-stop paste-over-start outfile
# All times measured in samples.
rate=`soxi -r "$1"`
e=`expr $rate '*' 5 / 1000` # Using default excess
l=$e                        # and leeway.
sox "$1" piece.au trim `expr $2 - $e - $1`s \
    `expr $3 - $2 + $e + $1 + $e`s
sox "$1" part1.au trim 0 `expr $4 + $e`s
sox "$1" part2.au trim `expr $4 + $3 - $2 - $e - $1`s
sox part1.au piece.au part2.au "$5" splice \
    `expr $4 + $e`s \
    `expr $4 + $e + $3 - $2 + $e + $1 + $e`s
```

In the above Bourne shell script, two splices are used to 'copy and paste' audio.

The SoX command

```
play "|sox -n -p synth 1 sin %1" "|sox -n -p synth 1 sin %3"
```

generates and plays two notes, but there is a nasty click at the transition; the click can be removed by appending **splice 1** to the command. (Clicks at the beginning and end of the audio can be removed by *preceding* the splice effect with **fade q .01 2 .01**).

\* \* \*

It is also possible to use this effect to perform general cross-fades, e.g. to join two songs. In this case, *excess* would typically be an number of seconds, and *leeway* should be set to zero.

**stat** [-s *scale*] [-rms] [-freq] [-v] [-d]

Display time and frequency domain statistical information about the audio. Audio is passed unmodified through the SoX processing chain.

The information is output to the ‘standard error’ (stderr) stream and is calculated, where  $n$  is the duration of the audio in samples,  $c$  is the number of audio channels,  $r$  is the audio sample rate, and  $x_k$  represents the PCM value (in the range  $-1$  to  $+1$  by default) of each successive sample in the audio, as follows:

<i>Samples read</i>	$n \times c$	
<i>Length (seconds)</i>	$n \div r$	
<i>Scaled by</i>		See -s below.
<i>Maximum amplitude</i>	$\max(x_k)$	The maximum sample value in the audio; usually this will be a positive number.
<i>Minimum amplitude</i>	$\min(x_k)$	The minimum sample value in the audio; usually this will be a negative number.
<i>Midline amplitude</i>	$\frac{1}{2} \min(x_k) + \frac{1}{2} \max(x_k)$	
<i>Mean norm</i>	$\frac{1}{n} \sum  x_k $	The average of the absolute value of each sample in the audio.
<i>Mean amplitude</i>	$\frac{1}{n} \sum x_k$	The average of each sample in the audio. If this figure is non-zero, then it indicates the presence of a D.C. offset (which could be removed using the <b>dcshift</b> effect).
<i>RMS amplitude</i>	$\sqrt{\frac{1}{n} \sum x_k^2}$	The level of a D.C. signal that would have the same power as the audio’s average power.
<i>Maximum delta</i>	$\max( x_k - x_{k-1} )$	
<i>Minimum delta</i>	$\min( x_k - x_{k-1} )$	
<i>Mean delta</i>	$\frac{1}{n-1} \sum  x_k - x_{k-1} $	
<i>RMS delta</i>	$\sqrt{\frac{1}{n-1} \sum (x_k - x_{k-1})^2}$	
<i>Rough frequency</i>		In Hz.
<i>Volume Adjustment</i>		The parameter to the <b>vol</b> effect which would make the audio as loud as possible without clipping. Note: See the discussion on <b>Clipping</b> above for reasons why it is rarely a good idea actually to do this.

The -s option can be used to scale the input data by a given factor. The default value of *scale* is

2147483647 (i.e. the maximum value of a 32-bit signed integer). Internal effects always work with signed long PCM data and so the value should relate to this fact.

The **-rms** option will convert all output average values to ‘root mean square’ format.

The **-v** option displays only the ‘Volume Adjustment’ value.

The **-freq** option calculates the input’s power spectrum (4096 point DFT) instead of the statistics listed above.

The **-d** option displays a hex dump of the 32-bit signed PCM data audio in SoX’s internal buffer. This is mainly used to help track down endian problems that sometimes occur in cross-platform versions of SoX.

#### **swap** [*l 2 | l 2 3 4*]

Swap channels in multi-channel audio files. Optionally, you may specify the channel order you would like the output in. This defaults to output channel 2 and then 1 for stereo and 2, 1, 4, 3 for quad-channels. An interesting feature is that you may duplicate a given channel by overwriting another. This is done by repeating an output channel on the command-line. For example, **swap 2 2** will overwrite channel 1 with channel 2; creating a stereo file with both channels containing the same audio.

See also the **remix** effect.

#### **stretch** *factor* [*window fade shift fading*]

Change the audio duration (but not its pitch). This effect is broadly equivalent to the **tempo** effect with (*factor* inverted and) *search* set to zero, so in general, its results are comparatively poor; it is retained as it can sometimes out-perform **tempo** for small *factors*.

*factor* of stretching: >1 lengthen, <1 shorten duration. *window* size is in ms. Default is 20ms. The *fade* option, can be ‘lin’. *shift* ratio, in [0 1]. Default depends on stretch factor. 1 to shorten, 0.8 to lengthen. The *fading* ratio, in [0 0.5]. The amount of a fade’s default depends on *factor* and *shift*.

See also the **tempo** effect.

#### **synth** [*len*] {[*type*] [*combine*] [[*%*]*freq*[*k*][:|+|/-[*%*]*freq2*[*k*]]] [*off*] [*ph*] [*p1*] [*p2*] [*p3*]}

This effect can be used to generate fixed or swept frequency audio tones with various wave shapes, or to generate wide-band noise of various ‘colours’. Multiple synth effects can be cascaded to produce more complex waveforms; at each stage it is possible to choose whether the generated waveform will be mixed with, or modulated onto the output from the previous stage. Audio for each channel in a multi-channel audio file can be synthesised independently.

Though this effect is used to generate audio, an input file must still be given, the characteristics of which will be used to set the synthesised audio length, the number of channels, and the sampling rate; however, since the input file’s audio is not normally needed, a ‘null file’ (with the special name **-n**) is often given instead (and the length specified as a parameter to **synth** or by another given effect that can has an associated length).

For example, the following produces a 3 second, 48kHz, audio file containing a sine-wave swept from 300 to 3300 Hz:

```
sox -n output.au synth 3 sine 300-3300
```

and this produces an 8 kHz version:

```
sox -r 8000 -n output.au synth 3 sine 300-3300
```

Multiple channels can be synthesised by specifying the set of parameters shown between braces multiple times; the following puts the swept tone in the left channel and adds ‘brown’ noise in the right:

```
sox -n output.au synth 3 sine 300-3300 brownnoise
```

The following example shows how two synth effects can be cascaded to create a more complex waveform:

```
sox -n output.au synth 0.5 sine 200-500 \
    synth 0.5 sine fmod 700-100
```

Frequencies can also be given as a number of musical semitones relative to ‘middle A’ (440 Hz) by prefixing a ‘%’ character; for example, the following could be used to help tune a guitar’s ‘E’ strings:

```
play -n synth sine %-17
```

**N.B.** This effect generates audio at maximum volume (0dBFS), which means that there is a high chance of clipping when using the audio subsequently, so in most cases, you will want to follow this effect with the **gain** effect to prevent this from happening. (See also **Clipping** above.)

A detailed description of each **synth** parameter follows:

*len* is the length of audio to synthesise expressed as a time or as a number of samples; 0=input-length, default=0.

The format for specifying lengths in time is hh:mm:ss.frac. The format for specifying sample counts is the number of samples with the letter ‘s’ appended to it.

*type* is one of sine, square, triangle, sawtooth, trapezium, exp, [white]noise, pinknoise, brown-noise; default=sine

*combine* is one of create, mix, amod (amplitude modulation), fmod (frequency modulation); default=create

*freq/freq2* are the frequencies at the beginning/end of synthesis in Hz or, if preceded with ‘%’, semitones relative to A (440 Hz); for both, default=%0. If *freq2* is given, then *len* must also have been given and the generated tone will be swept between the given frequencies. The two given frequencies must be separated by one of the characters ‘:’, ‘+’, ‘/’, or ‘-’. This character is used to specify the sweep function as follows:

- : Linear: the tone will change by a fixed number of hertz per second.
- + Square: a second-order function is used to change the tone.
- / Exponential: the tone will change by a fixed number of semitones per second.
- Exponential: as ‘/’, but initial phase always zero, and stepped (less smooth) frequency changes.

Not used for noise.

*off* is the bias (DC-offset) of the signal in percent; default=0.

*ph* is the phase shift in percentage of 1 cycle; default=0. Not used for noise.

*p1* is the percentage of each cycle that is ‘on’ (square), or ‘rising’ (triangle, exp, trapezium); default=50 (square, triangle, exp), default=10 (trapezium).

*p2* (trapezium): the percentage through each cycle at which ‘falling’ begins; default=50. exp: the amplitude in percent; default=100.

*p3* (trapezium): the percentage through each cycle at which ‘falling’ ends; default=60.

**tempo** [-q] *factor* [*segment* [*search* [*overlap*]]]

Change the audio tempo (but not its pitch). The audio is chopped up into segments which are then shifted in the time domain and overlapped (cross-faded) at points where their waveforms are most similar (as determined by measurement of ‘least squares’).

By default, linear searches are used to find the best overlapping points; if the optional **-q** parameter is given, tree searches are used instead, giving a quicker, but possibly lower quality, result.



*factor* gives the ratio of new tempo to the old tempo, so e.g. 1.1 speeds up the tempo by 10%, and 0.9 slows it down by 10%.

The optional *segment* parameter selects the algorithm's segment size in milliseconds. The default value is 82 and is typically suited to making small changes to the tempo of music; for larger changes (e.g. a factor of 2), 50 ms may give a better result. When changing the tempo of speech, a segment size of around 30 ms often works well.

The optional *search* parameter gives the audio length in milliseconds (default 14) over which the algorithm will search for overlapping points. Larger values use more processing time and do not necessarily produce better results.

The optional *overlap* parameter gives the segment overlap length in milliseconds (default 12).

See also **speed** for an effect that changes tempo and pitch together, and **pitch** for an effect that changes pitch without changing tempo.

**treble** *gain* [*frequency*[**k**] [*width*[**s**|**h**|**k**|**o**|**q**]]]

Apply a treble tone-control effect. See the description of the **bass** effect for details.

**tremolo** *speed* [*depth*]

Apply a tremolo (low frequency amplitude modulation) effect to the audio. The tremolo frequency in Hz is given by *speed*, and the depth as a percentage by *depth* (default 40).

Note: This effect is a special case of the **synth** effect.

**trim** *start* [*length*]

Trim can trim off unwanted audio from the beginning and end of the audio. Audio is not sent to the output stream until the *start* location is reached.

The optional *length* parameter tells the number of samples to output after the *start* sample and is used to trim off the back side of the audio. Using a value of 0 for the *start* parameter will allow trimming off the back side only.

Both options can be specified using either an amount of time or an exact count of samples. The format for specifying lengths in time is hh:mm:ss.frac. A start value of 1:30:5 will not start until 1 minute, thirty and ½ seconds into the audio. The format for specifying sample counts is the number of samples with the letter 's' appended to it. A value of 8000s will wait until 8000 samples are read before starting to process audio.

**vol** *gain* [*type* [*limitergain*]]

Apply an amplification or an attenuation to the audio signal. Unlike the **-v** option (which is used for balancing multiple input files as they enter the SoX effects processing chain), **vol** is an effect like any other so can be applied anywhere, and several times if necessary, during the processing chain.

The amount to change the volume is given by *gain* which is interpreted, according to the given *type*, as follows: if *type* is **amplitude** (or is omitted), then *gain* is an amplitude (i.e. voltage or linear) ratio, if **power**, then a power (i.e. wattage or voltage-squared) ratio, and if **dB**, then a power change in dB.

When *type* is **amplitude** or **power**, a *gain* of 1 leaves the volume unchanged, less than 1 decreases it, and greater than 1 increases it; a negative *gain* inverts the audio signal in addition to adjusting its volume.

When *type* is **dB**, a *gain* of 0 leaves the volume unchanged, less than 0 decreases it, and greater than 0 increases it.

See [4] for a detailed discussion on electrical (and hence audio signal) voltage and power ratios.

Beware of **Clipping** when the increasing the volume.

The *gain* and the *type* parameters can be concatenated if desired, e.g. **vol 10dB**.

An optional *limitergain* value can be specified and should be a value much less than 1 (e.g. 0.05 or 0.02) and is used only on peaks to prevent clipping. Not specifying this parameter will cause no limiter to be used. In verbose mode, this effect will display the percentage of the audio that needed to be limited.

See also **compand** for a dynamic-range compression/expansion/limiting effect.

### Deprecated Effects

The following effects have been renamed or have their functionality included in another effect; they continue to work in this version of SoX but may be removed in future.

**key** [-q] *shift* [*segment* [*search* [*overlap*]]]

Change the audio key (i.e. pitch but not tempo). This is just an alias for the **pitch** effect.

**pan** *direction*

Mix the audio from one channel to another. Use **mixer** or **remix** instead of this effect.

The *direction* is a value from -1 to 1. -1 represents far left and 1 represents far right.

**polyphase** [-w **nut**|**ham**] [-width *n*] [-cut-off *c*]

Change the sampling rate using ‘polyphase interpolation’, a DSP algorithm. **polyphase** copes with only certain rational fraction resampling ratios, and, compared with the **rate** effect, is generally slow, memory intensive, and has poorer stop-band rejection.

If the -w parameter is **nut**, then a Blackman-Nuttall (~90 dB stop-band) window will be used; **ham** selects a Hamming (~43 dB stop-band) window. The default is Blackman-Nuttall.

The -width parameter specifies the (approximate) width of the filter. The default is 1024 samples, which produces reasonable results.

The -cut-off value (*c*) specifies the filter cut-off frequency in terms of fraction of frequency band-width, also known as the Nyquist frequency. See the **resample** effect for further information on Nyquist frequency. If up-sampling, then this is the fraction of the original signal that should go through. If down-sampling, this is the fraction of the signal left after down-sampling. The default is 0.95.

See also **rate**, **rabbit** and **resample** for other sample-rate changing effects.

**rabbit** [-c0|-c1|-c2|-c3|-c4]

Change the sampling rate using libamplerate, also known as ‘Secret Rabbit Code’. This effect is optional and, due to licence issues, is not included in all versions of SoX. Compared with the **rate** effect, **rabbit** is very slow.

See <http://www.mega-nerd.com/SRC> for details of the algorithms. Algorithms 0 through 2 are progressively faster and lower quality versions of the sinc algorithm; the default is -c0. Algorithm 3 is zero-order hold, and 4 is linear interpolation.

See also **rate**, **polyphase** and **resample** for other sample-rate changing effects, and see **resample** for more discussion of resampling.

**resample** [-qs|-q|-ql] [*rolloff* [*beta*]]

Change the sampling rate using simulated analog filtration. Compared with the **rate** effect, **resample** is slow, and has poorer stop-band rejection. Only the low quality option works with all resampling ratios.

By default, linear interpolation of the filter coefficients is used, with a window width about 45 samples at the lower of the two rates. This gives an accuracy of about 16 bits, but insufficient stop-band rejection in the case that you want to have roll-off greater than about 0.8 of the Nyquist frequency.

The -q\* options will change the default values for roll-off and beta as well as use quadratic interpolation of filter coefficients, resulting in about 24 bits precision. The -qs, -q, or -ql options specify increased accuracy at the cost of lower execution speed. It is optional to specify roll-off

and beta parameters when using the **-q\*** options.

Following is a table of the reasonable defaults which are built-in to SoX:

Option	Window	Roll-off	Beta	Interpolation
(none)	45	0.80	16	linear
<b>-qs</b>	45	0.80	16	quadratic
<b>-q</b>	75	0.875	16	quadratic
<b>-ql</b>	149	0.94	16	quadratic

**-qs**, **-q**, or **-ql** use window lengths of 45, 75, or 149 samples, respectively, at the lower sample-rate of the two files. This means progressively sharper stop-band rejection, at proportionally slower execution times.

*rolloff* refers to the cut-off frequency of the low pass filter and is given in terms of the Nyquist frequency for the lower sample rate. *rolloff* therefore should be something between 0 and 1, in practise 0.8–0.95. The defaults are indicated above.

The *Nyquist frequency* is equal to half the sample rate. Logically, this is because the A/D converter needs at least 2 samples to detect 1 cycle at the Nyquist frequency. Frequencies higher than the Nyquist will actually appear as lower frequencies to the A/D converter and is called aliasing. Normally, A/D converts run the signal through a lowpass filter first to avoid these problems.

Similar problems will happen in software when reducing the sample rate of an audio file (frequencies above the new Nyquist frequency can be aliased to lower frequencies). Therefore, a good resample effect will remove all frequency information above the new Nyquist frequency.

The *rolloff* refers to how close to the Nyquist frequency this cut-off is, with closer being better. When increasing the sample rate of an audio file you would not expect to have any frequencies exist that are past the original Nyquist frequency. Because of resampling properties, it is common to have aliasing artifacts created above the old Nyquist frequency. In that case the *rolloff* refers to how close to the original Nyquist frequency to use a highpass filter to remove these artifacts, with closer also being better.

The *beta*, if unspecified, defaults to 16. This selects a Kaiser window. You can select a Blackman-Nuttall window by specifying anything  $\leq 2$  here. For more discussion of beta, look under the **filter** effect.

Default parameters are, as indicated above, Kaiser window of length 45, roll-off 0.80, beta 16, linear interpolation.

Note: **-qs** is only slightly slower, but more accurate for 16-bit or higher precision.

See also **rate**, **polyphase** and **rabbit** for other sample-rate changing effects. There is a detailed analysis of **resample** and **polyphase** at <http://leute.server.de/wilde/resample.html>; see **rabbit** for a pointer to its own documentation.

## DIAGNOSTICS

Exit status is 0 for no error, 1 if there is a problem with the command-line parameters, or 2 if an error occurs during file processing.

## BUGS

Please report any bugs found in this version of SoX to the mailing list ([sox-users@lists.sourceforge.net](mailto:sox-users@lists.sourceforge.net)).

## SEE ALSO

**soxi**(1), **soxformat**(7), **libsox**(3)

**audacity**(1), **ImageMagick**(1), **gnuplot**(1), **octave**(1), **wget**(1)

The SoX web site at <http://sox.sourceforge.net>

SoX scripting examples at <http://sox.sourceforge.net/Docs/Scripts>

**References**

- [1] R. Bristow-Johnson, *Cookbook formulae for audio EQ biquad filter coefficients*, <http://musicdsp.org/files/Audio-EQ-Cookbook.txt>
- [2] Wikipedia, *Q-factor*, [http://en.wikipedia.org/wiki/Q\\_factor](http://en.wikipedia.org/wiki/Q_factor)
- [3] Scott Lehman, *Effects Explained*, <http://harmony-central.com/Effects/effects-explained.html>
- [4] Wikipedia, *Decibel*, <http://en.wikipedia.org/wiki/Decibel>
- [5] Richard Furse, *Linux Audio Developer's Simple Plugin API*, <http://www.ladspa.org>
- [6] Richard Furse, *Computer Music Toolkit*, <http://www.ladspa.org/cmt>
- [7] Steve Harris, *LADSPA plugins*, <http://plugin.org.uk>

**LICENSE**

Copyright 1991 Lance Norskog and Sundry Contributors.

Copyright 1998–2008 Chris Bagwell and SoX Contributors.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

**AUTHORS**

Chris Bagwell (cbagwell@users.sourceforge.net). Other authors and contributors are listed in the AUTHORS file that is distributed with the source code.