

1 Student presentation (PA1)

- ████

2 Subject evaluation

3 Group picture

4 Why edit sound on the command line?

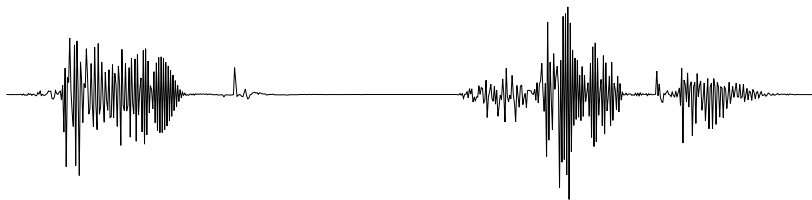



FIGURE 1. Graphical representation of sound 

- We are used to editing sound graphically.
- But for many operations, we do not actually need to *see* the waveform!

4.1 Potential applications

- | | |
|---------|---------|
| • _____ | • _____ |
| • _____ | • _____ |
| • _____ | • _____ |
| • _____ | • _____ |
| • _____ | • _____ |
| • _____ | • _____ |
| • _____ | • _____ |
| • _____ | • _____ |

4.2 Advantages

- No visual belief system (what you *hear* is what you hear)
- Faster (no need to load GUIs or waveforms)
- Efficient batch-processing (applying editing sequence to multiple files)
- Self-documenting (simply save an editing sequence to a script)
- Imaginative (might give you different ideas of what's possible)
- Way cooler (let's face it) ☺

4.3 Software packages

On Debian-based GNU/Linux systems (e.g., Ubuntu), install any of the below packages via apt, e.g., `sudo apt-get install mplayer`.

Program	.deb package	Function
mplayer	mplayer	Play <i>any</i> media file
sndfile-info	sndfile-programs	Metadata retrieval
sndfile-convert	sndfile-programs	Bit depth conversion
sndfile-resample	samplerate-programs	Resampling
lame	lame	MP3 encoder
flac	flac	FLAC encoder
oggenc	vorbis-tools	Ogg Vorbis encoder
ffmpeg	ffmpeg	Media conversion tool
mencoder	mencoder	Media conversion tool
sox	sox	Sound editor
ecasound	ecasound	Sound editor

TABLE 1. Command-line programs for playing, converting, and editing media files

4.4 Real-world examples

- *Silver Sounds* installation (Naughton Gallery Belfast, UK, 2007)
 - 10 submissions by different artists working on various platforms
 - File format conversions `.aif` to `.wav`
 - Resampling and bit depth conversions
 - MP3 encoding
- *24/7* sound installation (Ps² Gallery Belfast, UK, 2009)
 - 3 weeks of continuous ambience recordings (400+ hours; 236 GB)
 - Lossless compression to `.flac` reduces that to 78 GB
 - One line of code (plus one night of sleep):
`flac --delete-input-file *.wav`

5 Command line practice

5.1 Opening the command line

- Ubuntu (Unity): `Ctrl+Alt+t` (or type 'Terminal' in Dash)
- Mac os x Finder: Applications ▶ Utilities ▶ Terminal.app
- Windows: Start » All Programs » Accessoires » Command Prompt

5.2 Prompt

- Indicates that command line is ready for input
- Appearance varies between systems (and can be customized)

TABLE 2. Default command-line prompts on different operating systems

Os	Prompt
Ubuntu	user@host:~\$
Mac os x	host:~ user\$
Windows	C:\Windows\system32>

5.3 Executing (and interrupting) commands

- Commands executed with `↵` and return to prompt on completion:

```
host:~ user$ ls ↵
bla.txt foo.wav my.doc
host:~ user$
```

- If prompt does not return, command is probably still at work:

```
host:~ user$ sleep 2 ↵
host:~ user$
```

- Successful execution does not necessarily generate *any* printout!
- Terminate by force using `Ctrl+c` (careful when moving files!)
- Usually single command per line

5.4 Single command on multiple lines

- Split commands across multiple lines with backslash:

```
host:~ user$ sleep \ ↵
> 2 ↵
host:~ user$
```

5.5 Multiple commands on a single line

- Multiple commands can be sequenced with semicolons:

```
host:~ user$ sleep 2; ls ↵
bla.txt foo.wav my.doc
```

5.6 Unix command structure

- Follows pattern: `command[_flag][_value]_argument`
- Example: List (`ls`) all (`-a`) files in current directory (`.`):

```
host:~ user$ ls -a . ↵
```

- *White space* carries meaning! `ls_-a` ≠ `ls-a`
- Unix is *case sensitive*! `Desktop` ≠ `desktop`

5.7 Basic file system operations

Command	Meaning
<code>pwd</code>	Print working directory
<code>cd /path/to/target</code>	Change directory
<code>ls</code>	List current dir's contents
<code>ls -l</code>	More verbose <code>ls</code>
<code>ls -a</code>	Show also hidden files
<code>ls -lah</code>	Flags can be combined
<code>cp /path/to/source /path/to/target</code>	Copy source to target
<code>cp -r /path/to/dir /path/to/target</code>	Copy directory
<code>rm /path/to/file</code>	Remove file (for good!)
<code>rm -r /path/to/dir</code>	Remove dir (for good!)
<code>mv /path/to/source /path/to/target</code>	Move (rename) file or dir

TABLE 3. Key bindings for navigating within long commands

Linux	Mac os x	Go to...
<code>Ctrl+a</code>	<code>ctrl+a</code>	Start of line
<code>Ctrl+e</code>	<code>ctrl+e</code>	End of line
<code>Alt+f</code>	<code>⌘+f</code>	Next word
<code>Alt+b</code>	<code>⌘+b</code>	Prev. word

TABLE 4. Basic file system operations on the Unix command line

5.8 The need for speed

Think this is slow? Try the shortcuts from table 3 & 5 and think again!

Action	Meaning
<code>↑</code>	Go back in command history
<code>↓</code>	Go forward in command history
<code>→</code>	Auto-completion (turbo mode)
<code>Ctrl+r</code>	Recursive history search (super turbo mode)
<code>!cd</code>	Repeat last command that started with <code>cd</code>
<code>ls !*</code>	Repeat command (here: <code>ls</code>) with arguments from last call

TABLE 5. Gaining speed on the command line

5.9 Absolute vs. relative path notation

- *Absolute path notation* starts from root directory (i.e., with a slash)¹
- *Relative path notation* starts from current working directory (no slash)
- Example:

```
$ cd /Users/me ↵ (absolute)
$ pwd ↵
/Users/me
$ cd Desktop ↵ (relative)
$ pwd ↵
/Users/me/Desktop
```

- Several useful shorthands (cf., table 6):

```
$ cd /; pwd; cd ~; pwd; cd .; pwd; cd ..; pwd; cd - ↵
/
/Users/me
/Users/me
/Users
~
```

¹ To test the following commands, replace `me` with the output of the `whoami` command on your machine. On Linux, additionally replace `/Users` with `/home`.

TABLE 6. Synonyms for frequently used directories

Notation	Meaning
.	Current directory
..	Parent directory
/	Root directory
~	Current user's home dir
-	Previous dir (cd only)

6 Introduction to SoX

6.1 Installation & testing

1. Download and install latest version (14.4.2)

- Debian/Ubuntu: `$ sudo apt-get install sox ↵`
- Mac (with Homebrew):²
 - (a) Install Homebrew with Ruby command at <https://brew.sh/>
 - (b) Install SoX:³ `$ brew install sox ↵`
- Windows installer (use `.exe`, not `.zip`): <https://sourceforge.net/projects/sox/files/sox/14.4.2/sox-14.4.2-win32.exe>

2. Confirm SoX works:

```
↵
$ sox --version
```

Should print SoX version number (14.4.2)

3. Download example sound files from OCW page: MIT21M_380F16_sox_audio_files.zip

4. Unpack examples sounds to `sox_audio_files/`

² You can also download a `.zip` archive with binaries for `os x` directly from the SoX website (without installing Homebrew). However, you will then have to manually move the SoX binary to a directory in your system `$PATH`, in order to be able to execute `sox` on the command line without having to specify the path to the binary. Details are provided elsewhere in this document.

³ Or, if you want to use SoX with `.mp3`, `.flac`, or `.ogg` files, install it with the required libraries:
`brew install sox --with-lame --with-flac --with-libvorbis`

6.2 Getting help

- Built-in help: `$ sox --help ↵`
- Online documentation: <http://sox.sourceforge.net/Docs/Documentation>
- HTML manual: <http://sox.sourceforge.net/sox.html>
- PDF manual: <http://sox.sourceforge.net/sox.pdf>
- Mailing lists (low-volume):
http://sourceforge.net/mail/?group_id=10706

6.3 SoX command syntax

`/path/to/sox /path/to/in.wav /path/to/out.wav <fx1> <fx2> ...`

- Paths in absolute or (preferably) relative notation
- After issuing `cd /path/to/`, one can:
 - Write `in.wav` and `out.wav` without prepending `/path/to/` ☺
 - Write `./sox` (“in current directory”) instead of `/path/to/sox` ☺
- If `sox` binary is in system `$PATH`, one can write `sox` instead of `./sox` ⁴ ☺

6.4 Hello world!

Our first SoX edit: Reverberate `in.wav`, save result to `out.wav` & play

```
$ cd /path/to/sox_audio_files/ ↵
$ play in.wav ↵
$ sox in.wav out.wav reverb ↵
$ play out.wav ↵
```

⁴ This should be the case if you have installed SoX through `apt` (GNU/Linux) or `brew` (Mac OS X) or the `.exe` installer (Windows). On Linux or OS X, if in doubt, check whether the output of which `sox` appears in the colon-separated list of directories printed by `echo $PATH`.

7 SoX examples

7.1 Recording & playing sound

- Record 2 seconds of audio and play result:

```
$ rec foo.wav trim 0 2 ↵
$ play foo.wav ↵
```

- Doesn't work on Windows? Try this:⁵

```
$ sox -t waveaudio <device_number> foo.wav trim 0 2 ↵
$ sox foo.wav -t waveaudio <device_number> ↵
```

- Get information about recorded file (5 methods):

```
$ soxi foo.wav ↵ (soxi, not sox!)
$ soxi -r foo.wav ↵
$ soxi -t foo.wav ↵
$ sox foo.wav -n stat ↵ (sox, not soxi!)
$ sox foo.wav -n stats ↵
```

⁵ In this command, `<device_number>` will depend on your machine, but `0` will usually work.

7.2 Generating test signals

- Generate and play 3 s low-level sine sweep (500 Hz to 900 Hz):


```
$ sox -n out.wav synth 3 sine 500-900 vol 0.1 ↵
$ play out.wav ↵
```
- Generate and play 4'33" of silence (overwrites previous out.wav):


```
$ sox -n -r 48000 out.wav trim 0 4:33 ↵
$ play out.wav ↵
```

7.3 Level & phase adjustments

- Listen to input file first:


```
$ play in.wav ↵
```
- Reduce level by -6 dB = half gain (3 methods):


```
$ sox in.wav out.wav vol -6dB ↵
$ sox in.wav out.wav vol 0.5 ↵
$ sox -v 0.5 in.wav out.wav ↵
```
- Play output:


```
$ play out.wav ↵
```
- Test without writing to out.wav:


```
$ play in.wav vol -6dB ↵
```
- Negative gain factors additionally invert phase:⁶

```
$ sox in.wav out.wav vol -0.5 ↵
```
- Normalize to -3 dB peak level (do *not* append dB!) and confirm:⁷

```
$ sox in.wav out.wav norm -3 ↵
$ sox in.wav -n stats ↵
$ sox out.wav -n stats ↵
```

7.4 Cutting & splicing

- Time specified as hh:mm:ss.ms (redundant zeros can be omitted)
- Extract first second (trim <start> <duration>):


```
$ sox in.wav out.wav trim 0 1 ↵
```
- Extract seconds 0.8–1.4:


```
$ sox in.wav out.wav trim 0.8 0.6 ↵
```
- First 12 s, 1 s fade-in, 2 s fade-out


```
$ sox in.wav out.wav trim 0 12 fade 1 0 2
```

⁶ The phase inversion will not be audible when playing the resulting in isolation like here. However, you can visually compare the waveforms of in.wav and out.wav in a GUI audio editor such as *Audacity* (use a high zoom factor) to confirm the phase inversion has indeed been performed.

⁷ Compare the line starting Pk level dB in the output of the sox [...]wav -n stats commands for in.wav and out.wav.

7.5 Concatenating & mixing

- Listen to input files first:

```
$ play in1.wav in2.wav ↵
```

- Concatenate them to single file:

```
$ sox in1.wav in2.wav out.wav ↵
```

Or use splice effect for more sophisticated concatenations

- Mix them at equal levels (requires identical channel number):

```
$ sox -m in1.wav in2.wav out.wav ↵
```

7.6 Mono-to-stereo conversions

- Create a pseudo-stereo file from a single mono file (2 methods):

```
$ play mono.wav ↵
```

```
$ sox mono.wav pseudo_stereo.wav remix 1 1 ↵
```

```
$ sox mono.wav -c 2 pseudo_stereo.wav ↵
```

- Create a true stereo file from two mono files:

```
$ play left.wav right.wav ↵
```

```
$ sox -M left.wav right.wav true_stereo.wav ↵
```

Note that -M (merge) is different from -m (mix)!

7.7 Stereo-to-mono conversions

- Extract left channel from stereo file (2 methods):

```
$ sox stereo.wav left_channel.wav remix 1 ↵
```

```
$ sox stereo.wav -c 1 left_channel.wav mixer -l ↵
```

- Extract right channel from stereo file (2 methods):

```
$ sox stereo.wav right_channel.wav remix 2 ↵
```

```
$ sox stereo.wav -c 1 right_channel.wav mixer -r ↵
```

- Mix stereo down to mono (3 methods):

```
$ sox stereo.wav mono_mixdown.wav remix 1,2 ↵
```

```
$ sox stereo.wav mono_mixdown.wav remix 1-2 ↵
```

```
$ sox stereo.wav -c 1 mono_mixdown.wav mixer 0.5,0.5 ↵
```

7.8 Swap stereo channels

- Swap L & R channels of a stereo file:

```
$ sox stereo.wav stereo_swapped.wav swap ↵
```


7.9 Sample rate conversion

- Convert to 8 kHz (2 methods):

```
$ sox in.wav out.wav rate 8k ↵
$ sox in.wav -r 8k out.wav ↵
```

7.10 Miscellaneous effects

- Reverse playback:

```
$ play in.wav reverse ↵
```

- Low-pass filter:

```
$ play in.wav lowpass 440 ↵
```

- Reverberate (append 2 sec of silence first to avoid cutting off decay):

```
$ play in.wav pad 0 2 reverb ↵
```

- Led-Zepelinesque reverse echo:

```
$ play in.wav reverse pad 0 1 reverb reverse ↵
```

- Chorus with arguments (check SoX manual for details):

```
$ play in.wav chorus 0.6 0.9 500.0 0.4 0.25 2.0 -t 60.0 \ ↵
0.32 0.4 1.3 -s ↵
```

- Multiple and single echoes:

```
$ play in.wav echos 0.4 0.6 400.0 0.5 900.0 0.3 ↵
$ play in.wav echo 0.7 0.89 1000.0 0.1 ↵
```

- A sequence of processing operations:

```
$ sox in.wav out.wav highpass 500 rate 96k norm -12 \ ↵
dither ↵
```

7.11 Noise reduction

- Listen to noisy input and isolated noise sample:

```
$ play noisy.wav ↵
$ play background_noise.wav ↵
```

- Step 1: Create noise profile:

```
$ sox background_noise.wav -n trim 0 1 noiseprof \ ↵
noise_profile ↵
```

- Step 2: Denoise (0.3 is a denoise factor 0...1):

```
$ sox noisy.wav denoised.wav noised noise_profile 0.3 ↵
```

- Listen to denoised result:

```
$ play denoised.wav ↵
```

- Or, as a single command using | (the 'pipe'):

```
$ sox background_noise.wav -n trim 0 1 noiseprof | play \
noisy.wav noised ↵
```

8 Shell scripts (GNU/Linux & Mac OS X only)

Any command sequence can be turned into a *shell script* for re-use. ☺

8.1 Example script

```
1 #!/bin/sh
2
3 # Above line: "Execute with Unix shell"
4
5 # Comments start with hash (#)
6
7 # Command-line printout
8 echo "Called $0 with $# arguments..."
9 echo "Converting $1 to $2..."
10
11 # Actual sound processing in SoX
12 sox $1 $2 reverse pad 0 1 reverb reverse
13
14 exit 0 # Indicates successful execution
```

LISTING 1. `zeppelinify.sh` shell script to generate reverse echo in the style of Led Zeppelin

- Save above code to *plain* text file `zeppelinify.sh`⁸
- Disable rich text formatting if you use MS Word or OS X Text Editor
- See table 7 for meaning of \$ placeholders

⁸ The `.sh` file extension is commonly used for shell scripts.

Placeholder	Meaning
<code>\$#</code>	Number of arguments passed to script
<code>\$0</code>	Name of script (including path)
<code>\$1</code>	First argument passed to script
<code>\$2</code>	Second argument passed to script

TABLE 7. Placeholders in shell scripts

8.2 Make script executable

- Make script executable:
\$ `chmod +x /path/to/zeppelinify.sh` ↵
- Execute script (`/path/to/in.wav` *must* exist):
\$ `/path/to/zeppelinify.sh /path/to/in.wav \` ↵
 `/path/to/out.wav` ↵
- Throws “Permission denied” error on os x? Try:
\$ `cd /path/to/` ↵
followed by one of the following two commands:
\$ `./zeppelinify.sh /path/to/in.wav /path/to/out.wav` ↵
\$ `sh zeppelinify.sh /path/to/in.wav /path/to/out.wav` ↵

8.3 Make script available system-wide

- Move script to a directory included in colon-separated list printed by:
\$ `echo $PATH` ↵
 `/usr/local/bin:/usr/bin:/bin:...`
- E.g., move `zeppelinify.sh` from current location to `/bin/zeppelinify`:⁹
\$ `sudo mv /path/to/zeppelinify.sh /bin/zeppelinify` ↵
- Test from home directory:
\$ `cd ~` ↵
\$ `zeppelinify` ↵

⁹ Note that such system-wide binaries are typically used *without* the `.sh` file extension. Also, `sudo` (“do as superuser”) is required here for write permissions to the system directory `/bin`.

8.4 Exercise: SoX m/s decoder script

Write an `m/s` decoder `ms2lr` in SoX, which can, for example, be called as

```
$ ms2lr ms.wav lr.wav ↵
```

- `ms.wav` ... existing `m/s`-encoded file (*M* on ch. 1 & *S* on ch. 2)
- `lr.wav` ... resulting decoded stereo file (*L* on ch. 1 & *R* on ch. 2)
- But user should be able to specify *arbitrary* input and output file names
- Bonus: Abort with error message if called with < 2 arguments

References & further reading

SoX developers (Dec. 31, 2014). *SoX. Sound eXchange, the Swiss Army knife of audio manipulation*. User manual. URL: <http://sox.sourceforge.net/sox.pdf> (visited on 02/27/2017).

MIT OpenCourseWare
<https://ocw.mit.edu/>

21M.380 Music and Technology: Recording Techniques and Audio Production
Fall 2016

For information about citing these materials or our Terms of Use, visit: <https://ocw.mit.edu/terms>.