SOUND SOFTWARE: TOWARDS SOFTWARE REUSE IN AUDIO AND MUSIC RESEARCH

Chris Cannam, Luís A. Figueira and Mark D. Plumbley
Centre for Digital Music, Queen Mary University of London
info@soundsoftware.ac.uk

soundsoftware.ac.uk

About us

Audio and music researchers...
- Come from many different backgrounds: electronics, computer science, music, dance...
- Don't always want to "be software engineers"
- Use many different languages and environments

SoundSoftware.ac.uk:
- We provide a service for the UK audio and music research community
- We aim to support development and reuse of software and data
- Our goal is to enable high quality research

Reproducible Research

Methods now so complex that the paper is not enough to reproduce results
Hence *Reproducible Research*: provide *article + data + software*

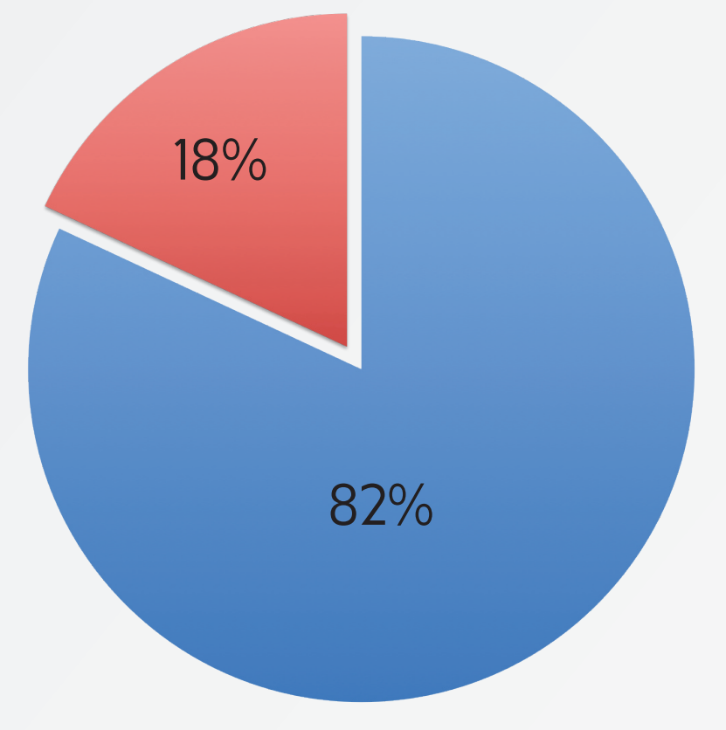
Growing interest in the community:
- Special session at ICASSP (2007)
- Special issue of IEEE *Signal Processing* magazine (2009)
- "The case for open computer programs" in *Nature* (2012)
- Policy changes, e.g. UK EPSRC funding body requiring data management plan (2012)

Yet take-up in the audio and music research field has been limited.
Why?

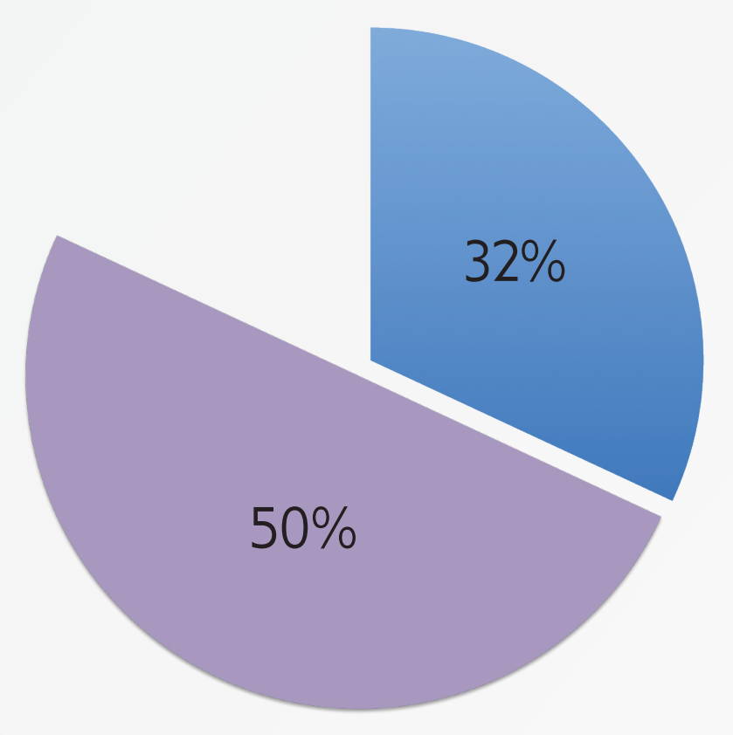
Real-World Limitations

Our survey of UK audio and music researchers
- October 2010 to April 2011
- 16 different institutions
- More than 70 responses

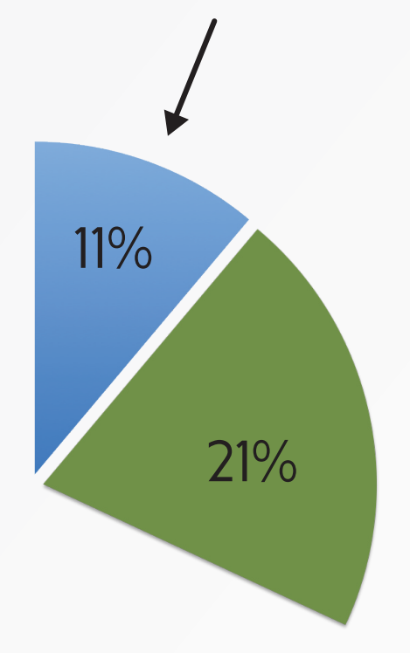
We found:
- Most researchers develop software
- Many take some steps toward reproducibility
- ... but very few have actually published code!



Above: 82% of respondents said they developed code during work...



...of whom 39% reported taking steps to reproducibility...



... of whom only 35% (that's 11% of the whole) reported publishing any of their software

"Little [of what I develop] is good enough for the public"
"I often give people my software... but do not publish or publicize it"
"Am currently looking for suitable version control"
"Not possible due to practical limitations, especially time"
"Data published but not full source"
"Not yet got round to it"
"I'm too protective of my work"
"Significant investment of additional time... cannot be justified"

Towards Sustainable Software

We aim to identify common practical barriers to publication and reuse, and propose straightforward "bottom-up" approaches to solving them

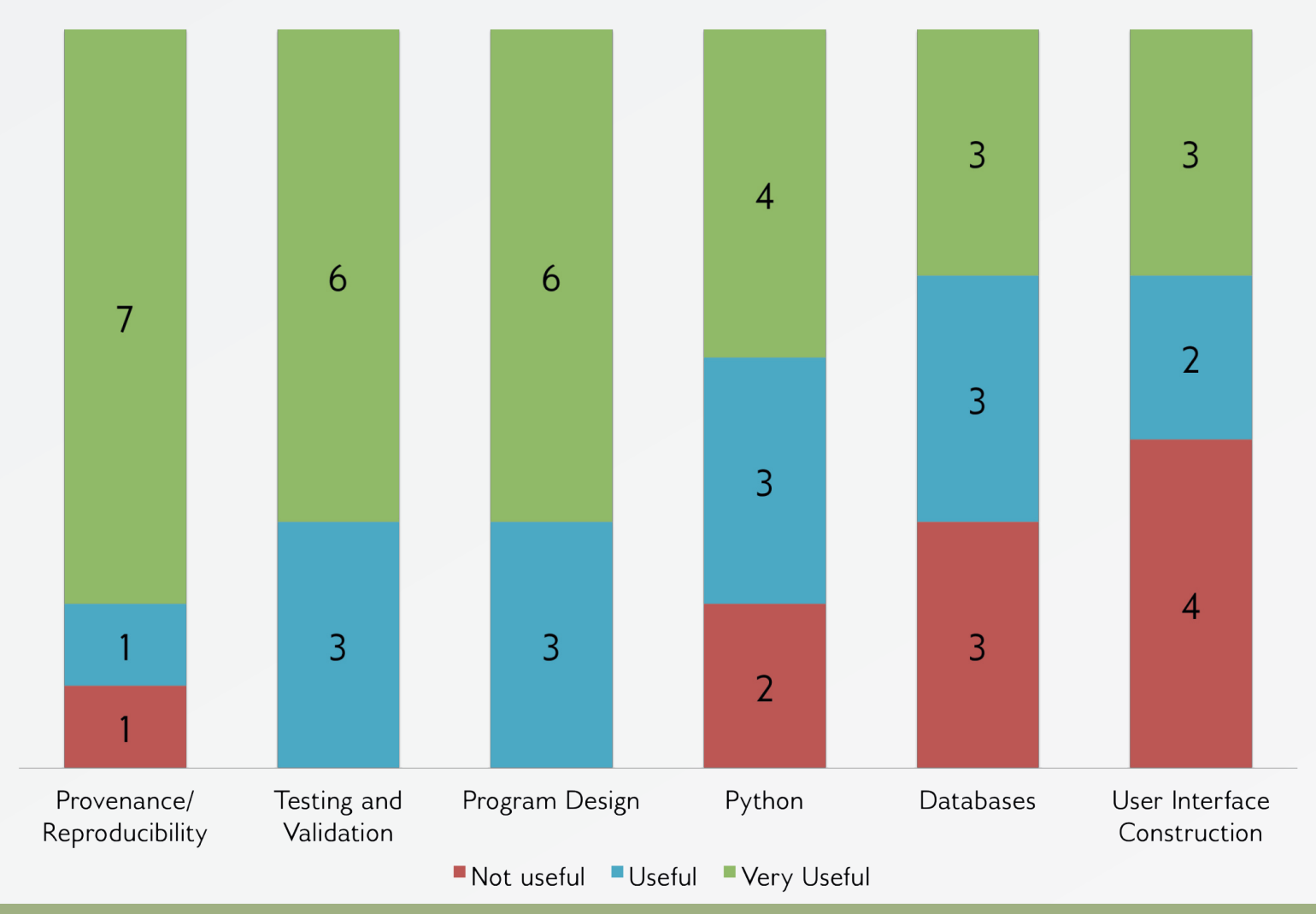
Barrier: Lack of education and confidence with code

Researchers are largely self-trained software developers: even simple approaches to training in testing and provenance can pay off

Autumn School:
- Based on Software Carpentry
- 20 researchers from UK institutions
- Week-long workshop
- Python, version control, testing etc

Future: Shorter workshops, in more places

Right: Post-workshop survey: "How useful did you find each topic for your research?"



	Provenance/ Reproducibility	Testing and Validation	Program Design	Python	Databases	User Interface Construction
🟩 Very Useful	7	6	6	4	3	3
🟦 Useful	1	3	3	3	3	2
🟥 Not useful	1			2	3	4

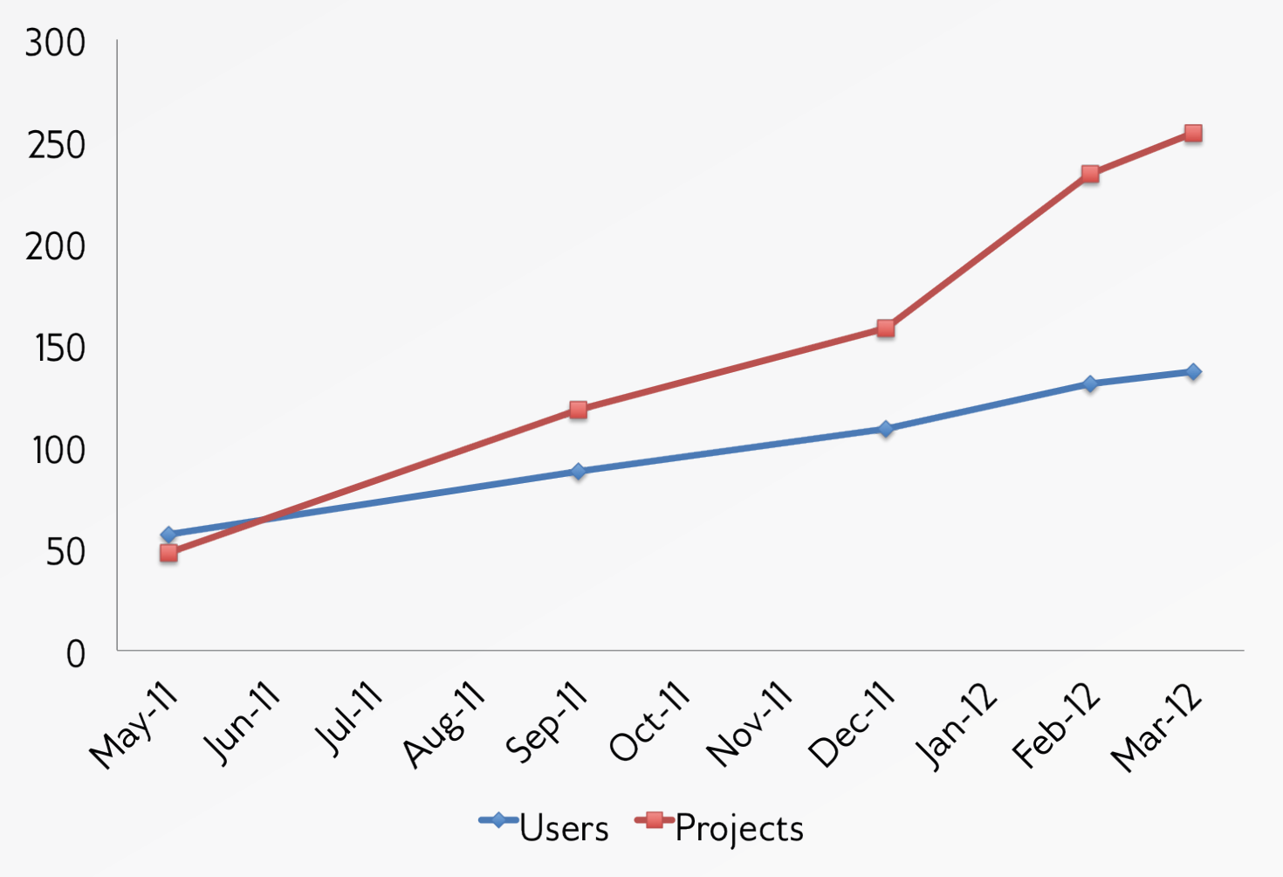
Barrier: Lack of facilities and tools

Only a minority of researchers use version control or develop collaboratively, and many institutions have no facilities available

Software code project site
- http://code.soundsoftware.ac.uk
- Version control with Mercurial
- Public and private projects

EasyMercurial
- Simple, cross platform version control
- Easy to teach, easy to learn

Right: Evolution of numbers of registered users and projects on our code site



Barrier: Lack of incentive for publication

Software outputs often not recognised, not cited, or not properly credited

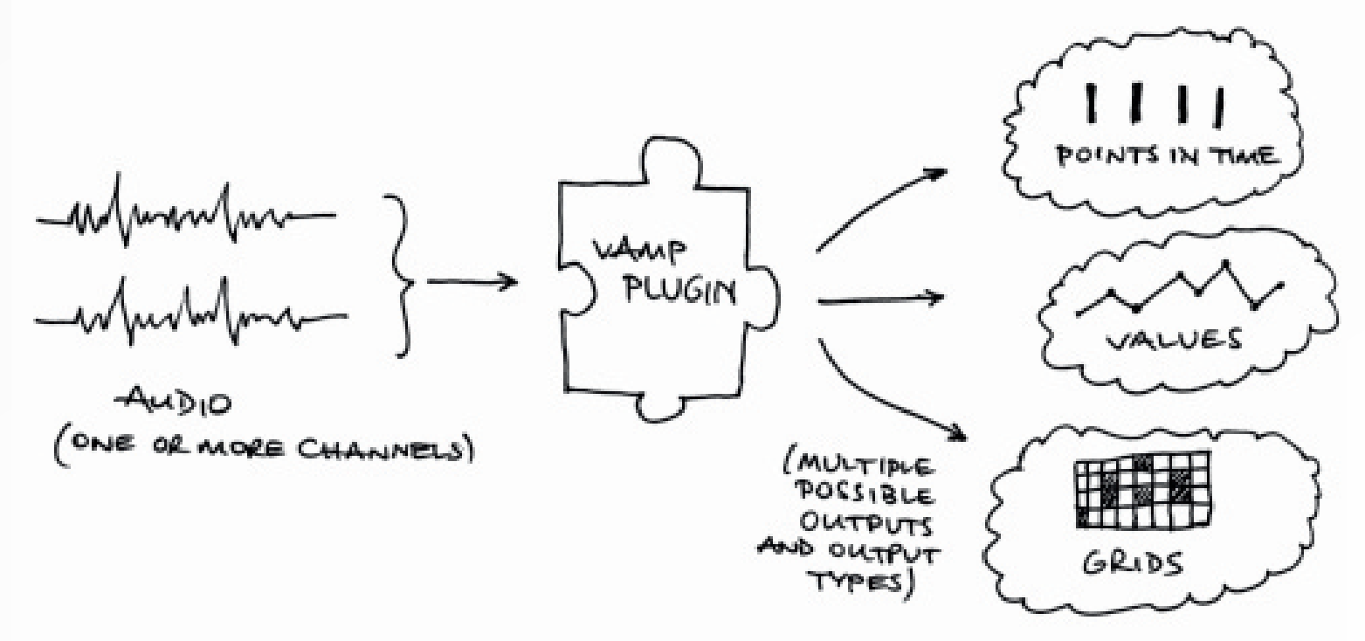
Associate publications with code in the code project site
- Make sure researchers can find the code, and code users can cite the publications

Barrier: Platform incompatibilities

Code written in many different languages and frameworks for many platforms; published code often not readily available to future researchers

Plugin approach
- Minimal route to usable software
- Avoid custom user-interfaces
- Attach to existing software ecosystems

Right: The Vamp plugin system, integrating research code into widely-available plugins (http://vamp-plugins.org)



Recommendations

Aim at easy targets

Basic software engineering techniques can make a big difference
Many researchers are self-taught in software and fear writing code
In this technical field, introductory training can provoke independent learning

Provide version control and software hosting services

Version control systems bring self-assurance and confidence
Make them available and encourage researchers to use them
Also useful for other tasks, such as writing articles with LaTeX

Turn code into plugins

Seek ways to integrate research code with end-user applications
Exploit existing ecosystems to disseminate and maintain software
Benefit from faster and easier deployment and a larger user-base

Encourage collaborative development

Create an environment of confidence about sharing and reusing code
Provide new opportunities to learn
We collaborate on papers, why not on software?

Conclusions and Future Work

Reproducible Research is hard, and especially hard to "add after publication"
We can make it easier through a bottom-up approach:
- Identify practical barriers, such as limitations in skills and tools
- Find simple measures against those problems: apply them from the start
- Prepare ground in which open publication can naturally grow

We need to evaluate the results of our work in terms of the proportion of publications with published and reusable code.

We plan further workshops, on a smaller scale and with broader distribution.

We plan to produce more training material for version control and our code site, and to evaluate how researchers respond to training with these facilities.

Read the full paper at
http://soundsoftware.ac.uk/icassp-2012-paper

EPSRC
Pioneering research and skills

This initiative is funded by grant EP/H043101/1 from the UK Engineering and Physical Sciences Research Council (EPSRC)

Queen Mary
University of London