soundsoftware.ac.uk

SoundSoftware: Towards Reusable Software for Audio & Music Research

Mark Plumbley

Centre for Digital Music

Queen Mary, University of London

Gatsby Institute

25 July 2014

EPSRC
Engineering and Physical Sciences Research Council

Overview

- Reproducible research
- Survey
- Research Pipeline problems
- Barriers and approaches
- Software Carpentry Boot Camps
- Reproducible Research Prizes
- D-CASE Data Challenge
- Conclusions

Reproducible Research

Reproducible Research

(Buckheit & Donoho, 1995; Vandewalle et al, 2009)

Idea: researchers should be able to reproduce the work of others.

Research used to be "reproducible" from the paper alone.

In audio & music research, methods are now too complex.

The paper is not enough: need algorithm, parameters, datasets, ...

So, we need

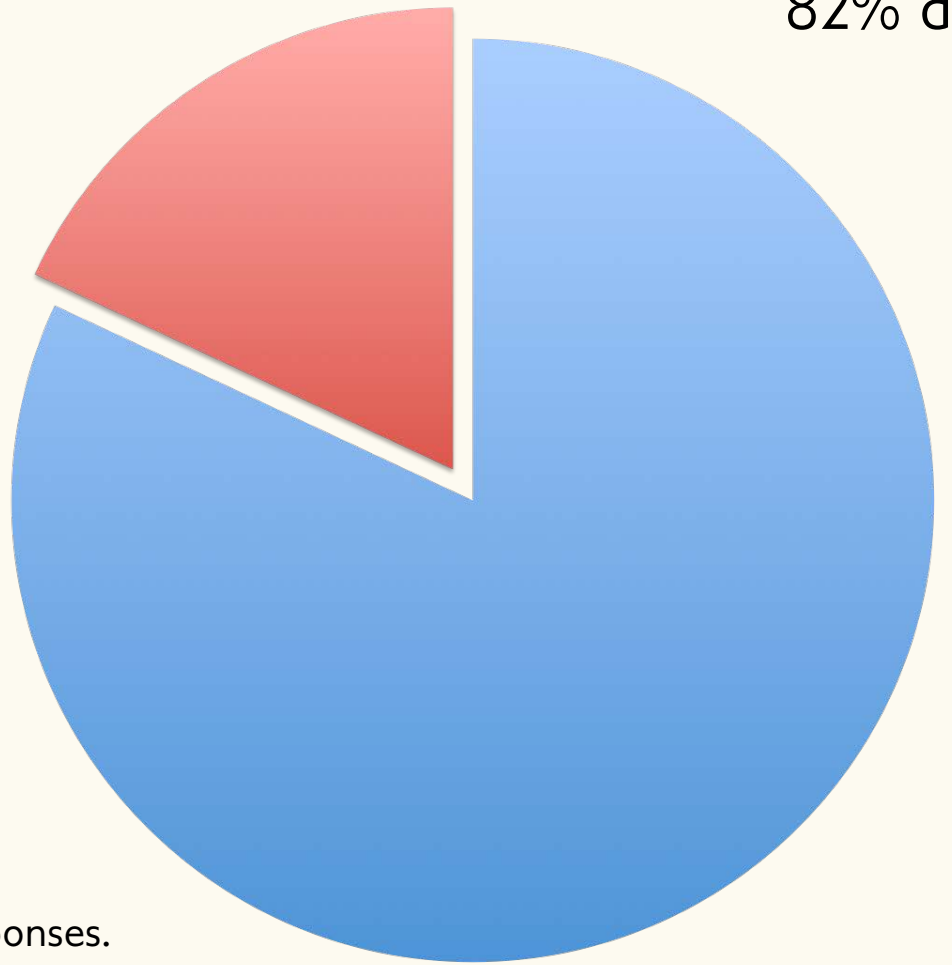
- The paper (ideally Open Access)
- The code (ideally Open Source)
- The data (ideally Open Data)

Well-known example: WaveLab (Buckheit & Donoho, 1995)

But in audio & music research, few people do this. Why?

Survey 2010–2011

We asked some researchers:*

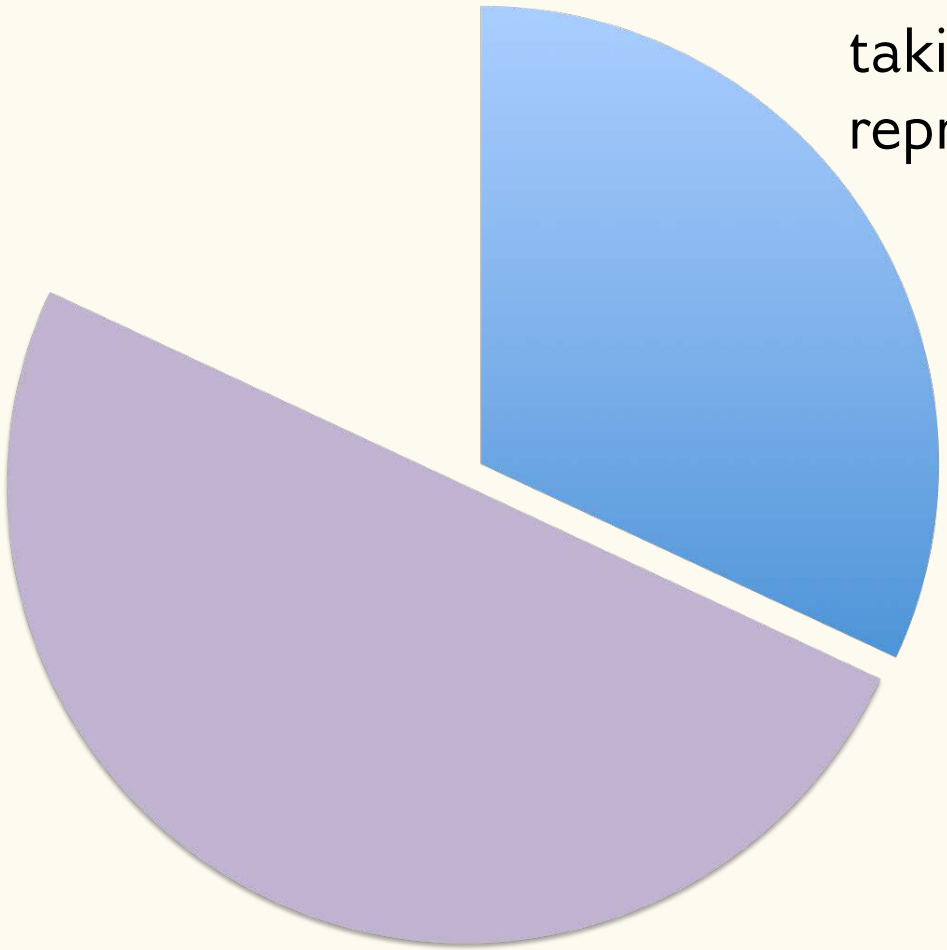


82% develop code

* - Oct 2010-Apr 2011,
54 complete + 23 partial responses.
For these figures we considered 72 responses.

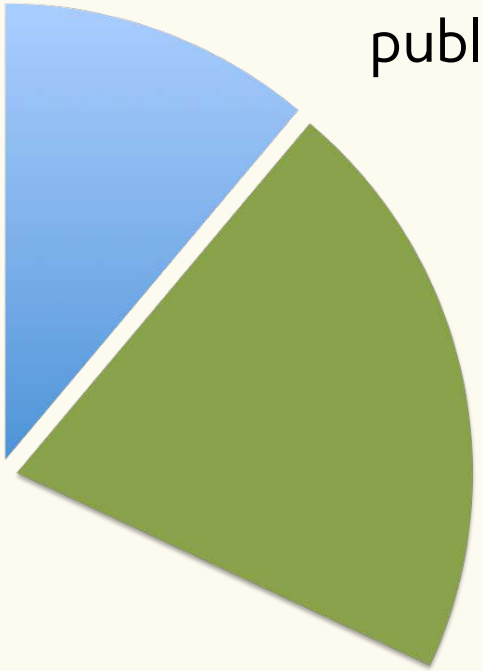
Survey 2010–2011

of whom 39% report taking steps to reproducibility



Survey 2010–2011

of whom 35% report publishing any code



Survey 2010–2011

That's 11% of the whole

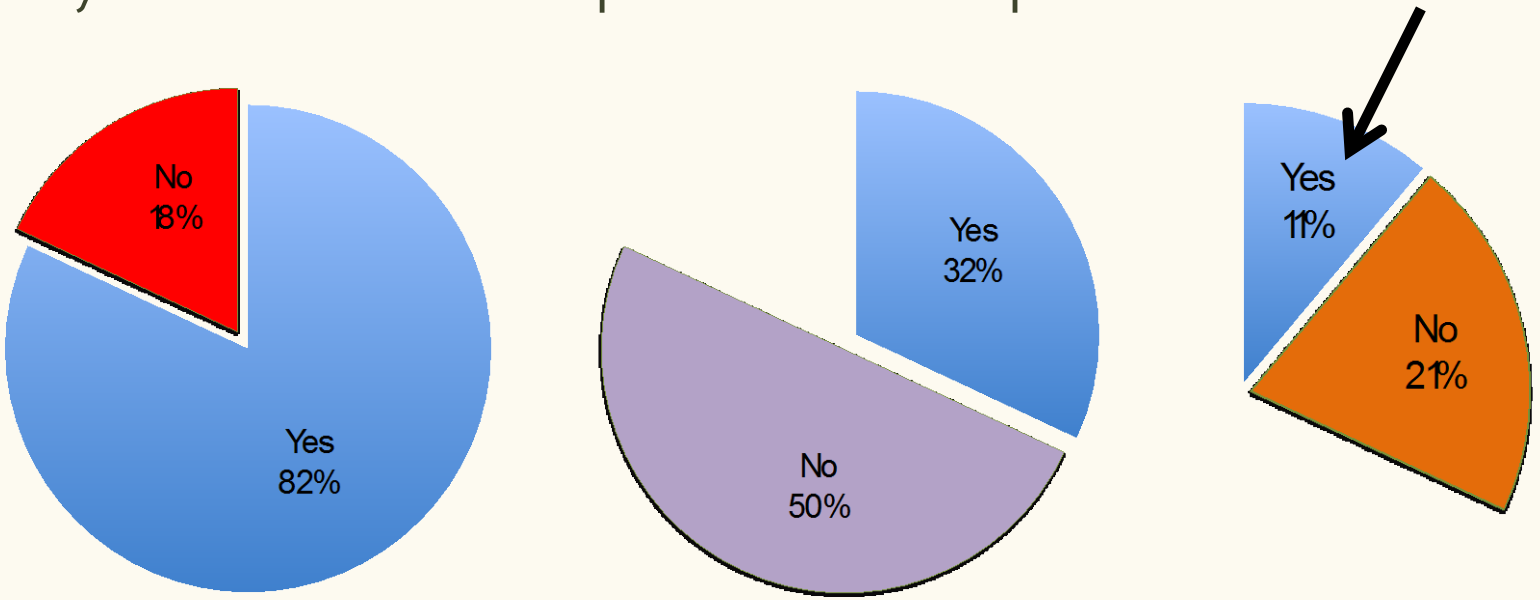
Summary

82% of respondents reported developing software, but

only 39% of those said they took steps to reproducibility

only 35% of *those* reported ever publishing any code

i.e. only **11%** tried to be reproducible and published the code.



Also: 51% said their code never left their own computer

Why don't people publish code?

We found:

- Lack of time
- Copyright restrictions
- Potential for future commercial use

Other factors (UK Research Information Network, 2010):

- Lack of evidence of benefits
- Culture of independence or competition
- Quality concerns (self-taught programmers)

Also: it takes effort early in the research cycle;
hard to find time/motivation after the paper is published

So instead of this Research Pipeline,

Researcher A ("Producer")

- Read background papers
- Do own research
- Publish paper X

Researcher B ("Consumer-Producer")

- Read paper X
- Understand/reproduce results in paper X
- Do more research building on X
- Publish paper Y that cites X / produce product that uses X

... and so on.

... we have: *Real* Research Pipeline

Researcher A ("Producer")

- Read background papers
- Do own research (including lots of coding)
- Publish paper X (not enough space for all the code)

Researcher B ("Consumer-Producer")

- Read paper X
- Can't reproduce or use results in paper X
- Tear out hair
- Give up / do something else

NB: A and B may be in same group (or same person later!)

Sustainability

Can still be problems, even if software is published...

- Software designed for legacy platforms / not maintained (Sun SPARC, NeXTSTEP, Dec Alpha)
- PhD students graduate, staff move:
  -> web pages containing original software/data lost
- Software/datasets in slightly different versions (error corrections, enhancements)

Other issues

- Copyright issues of datasets
  (e.g. audio of Beatles tracks cannot be placed on the web)

Example: Beat Tracking

- Classic algorithm (Scheirer, 1998) in legacy C++ for DEC Alpha, original website lost, various modified versions "passed around"
- Another early algorithm (Goto, 1994-8) written for a parallel architecture, computer no longer exists, code never released
- UK algorithm (Hainsworth, 2005) in Matlab but with Windows-only DLL component (limits its portability)
- Another key algorithm (Klapuri et al, 2006) in Matlab, available from authors, requires contract preventing redistribution
- Several other algorithms never released, but researchers may help individually (Cemgil, 2000; Laroche 2003; Peeters 2005).

So; hard for new researchers to compare with those algortihms!

Example of potential: Beat Tracking

- Onset detection (Bello, 2003) in beat tracking (Davies, 2005)
- Davies (Matlab), ported to SuperCollider by Collins (2006)
- Davies alg ported into cross-platform C++ Vamp plugin for Sonic Visualiser / Sonic Annotator (on EPSRC OMRAS2 project)
- Inspired Max/MSP beat tracking system (Robertson, 2007)
- Used in beat-synchronous audio effects (Stark, 2008) - developed in Matlab, ported to real-time VST plugins.
- Rhythm morphing (Hockman & Davies, 2008) originally Matlab, ported into a C++ library (on EPSRC Platform Grant)

Helped by (a) personal continuity, (b) funding for “extra step”

How can we tackle these issues?

We're taking a **bottom-up approach**:

- Make **incremental improvements** to development practice

  by

- Identifying **specific barriers** to publication and reuse, that are relatively straightforward to address

So we hope to:

- Increase perception among researchers that code is something you can work on together, that can be reused
- Prepare the ground for reproducible publication

Barriers to publication and reuse

- Lack of education and confidence with code
- Lack of facilities and tools
- Lack of incentive for publication
- (Also: Platform incompatibilities)

Education and training

Barrier: Lack of confidence in code

Issue: Researchers largely self-trained in software development

Our approach: Training in research software development

Relatively small amounts of training can pay off

Autumn School (Nov 2010, 5 days) based on Software Carpentry

- Version control systems
- Unit testing, test-driven development
- Python syntax and structure
- Managing experimental datasets

Software Carpentry Boot Camps

- Pre-conference Boot Camp (DAFx 2012, York, UK, pictured)
- Tutorials at conferences (DAFx 2012, ISMIR 2012)
- Integration into PhD training programme

Conference Tutorials

DAFx 2012 (York)

Introduction to issues around reproducible software, data management, open access etc

ISMIR 2012 (Porto)

Hands-on tutorial on unit-testing research software with Python

AES 53rd Conference 2014 (London)

Hands-on tutorial on building and structuring research code in the IPython Notebook using version control

Learning Resources

Handouts published on

- Publishing your research code
- Publishing your research data
- Getting started with the SoundSoftware code hosting site
- Choosing a software licence
- Unit testing

Coming soon: Accumulated suggestion packs for research groups

Facilities and tools

Barrier: Lack of facilities and tools

Issue: Researchers don't use code hosting / version control

- Research groups / institutions often do not provide any
- Researchers often unaware of them

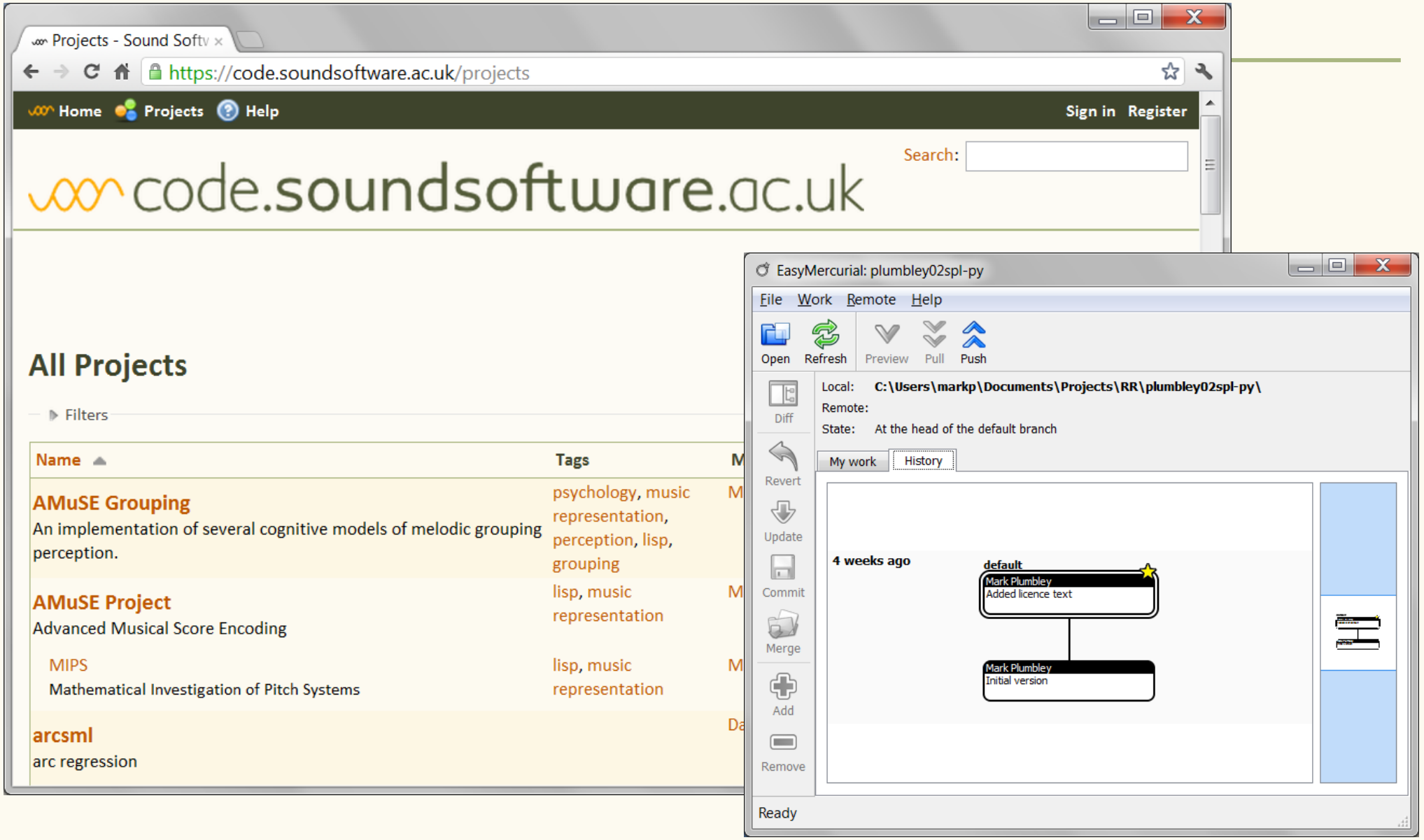
Our approach: code site: **http://code.soundsoftware.ac.uk**

- Focus on audio and music research
- Public and private projects
- Link publications with code

Also: User interfaces for version control

- Existing ones are surprisingly difficult

EasyMercurial: **http://easyhg.org** and tutorials and videos



Currently about 340 users and over 700 projects

Reproducible Research Repository

Somewhere to record "experiments" tying together publications, code, and datasets.

Hosts only the experimental record:

- What software and data was used and where to get it
- How the experiment was carried out
- Where it was published and what figures it produced

Anyone can add records, not just the original author!

Alpha version: Framework + initial experiments

→ http://rrr.soundsoftware.ac.uk/

Software

Code collaborations

SMALLbox sparse representation toolbox

- Available from SoundSoftware code site, c. 10K downloads

MPEG-M and MPEG-A (OCTV, IMAF) collaborations

- Including code and services support

Auditory Image Models

- Hosting and coordination

MATLAB-powered web applications, with ISVR Southampton

- (To be written up shortly!)

Tony pitch-annotation program, with NYU

- Working to a 1.0 release...

Plugin projects

Vamp plugin collaborations:

- Chordino and Segmentino
- Cepstral library and pitch tracker
- Constant-Q plugin and library
- Silvet note transcription (release coming soon!)

+ supporting tools such as the Vamp Test Plugin and Java Vamp interface

Engagement

Barrier: Lack of incentive

Issue: Software not recognised as valued research output

- Lacks publication conventions for authorship, academic rewards unclear

Approach (1): Link publications to code on the code site

- Increase likelihood of code users discovering your papers
- Ensure users know how to cite your work
- Increase take-up / impact of your research

Approach (2): Other encouragements

- Reproducible Research Prizes
- Data challenges

Reproducible Research Prizes

RR Prizes: Motivation

- Promote development and release of sustainable and reusable software associated with published research
- Recognise researchers who take the extra step, or whose work which will enable others to do so
- Offer a really clear short-term incentive

RR Prizes: What we did (and why)

- Broad call for submissions (April 2013)
- journal or conference papers, published or pending
- Very little idea what response we might get

"If you have published your software or datasets as part of your audio or music research output, so that other UK researchers can reproduce your results, you could win a prize!"

RR Prizes: Categories and prizes

Journal paper: New submission

Conference paper: New submission

Journal paper: Already published

Conference paper: Already published

Prizes: Article Processing Charge for open access publication; travel bursaries; other options

RR Prizes: Categories and prizes

Journal paper: New submission – one entry!

Conference paper: New submission – 3 entries

Journal paper: Already published – 3 entries

Conference paper: Already published – 5 entries

13 entries total (7 from UK) across 10 institutions

5 MATLAB; 3 C/C++; 2 Python; 1 Lisp

Two with no software (datasets only)

RR Prizes: Work and meta-work

Five traditional research papers

Two papers presenting “challenges”
Two presenting software applications
Two reviewing reproducibility of other work
One presenting a newly-compiled test dataset
One presenting a data interchange format

RR Prizes: Judging criteria

- Ease of reproducibility of the results

  → assessed by SoundSoftware (that's us)

- Quality of sustainability planning

  → assessed by the Software Sustainability Institute

- Potential to enable high quality research in the UK audio and music research community

  → assessed by external reviewers

How reproducible were they?

Somewhat... with a number of fiddly details!

- Hard-coded paths for dependency files and scripts (/home/myname/test1/data.csv)
- MATLAB version incompatibilities, missing Python modules
- Public datasets gone missing (e.g. Magnatagatune)
- Randomised test datasets, random matrix initialisers, etc

Good practices:

- Publishing via e.g. github or our own code site (5 submissions)
- Script to test the environment is set up correctly (1 submission)
- Scripts as used when assembling the actual paper!

Prize winners

Winners announced at SSW workshop June 2013

Majdak, P., et al, *Spatially Oriented Format for Acoustics: A Data Exchange Format Representing Head-Related Transfer Functions*

Sturm, B. L., et al, *Comments on "Automatic Classification of Musical Genres Using Inter-Genre Similarity"* – and two other papers

Giannoulis, D., et al, *A Database and Challenge for Acoustic Scene Classification and Event Detection*

Raffel, C., and Ellis, D., *Reproducing Pitch Experiments in "Measuring the Evolution of Contemporary Western Popular Music"*

Following rounds

- Linked to Audio Engineering Society (AES)
  53rd Conference on Semantic Audio (London, January 2014)
  - Prize submission deadlines coordinated with the AES
  - Hints for reproducible publication available
  - Encourage to think about software as they write paper
- Current: Link to IEEE MLSP 2014 (Reims, France, Sept 2014)

- Future: “RR Prize Kit” for conference organizers

D-CASE Challenge: Detection & Classification of Audio Scenes & Events

- Challenge in conjunction with IEEE AASP TC
- Acoustic Scene Classification:
  - Characterize the acoustic environment by semantic label
  - 10 classes: bus, busystreet, office, openairmarket, ...
- Acoustic Event Detection:
  - Detect and classify acoustic (potentially overlapping) events
  - 16 event types: cough, doorslam, drawer, keyboard, keys, knock, laughter, mouse, ...
- People submitted software to us.

D-CASE Challenge Summary

Outcome from WASPAA Special Session (12 posters)

- Lots of interest in the Challenge
- Many discussions about evaluation!

- Code available for 6 submitted systems:
  - Scene Classification: - Chum et al., - Olivetti
    - Geiger et al., - Roma et al.
  - Event Detection: - Gemmeke et al. - Vuegen et al.
  - Also: Our own baseline systems

More at: http://c4dm.eecs.qmul.ac.uk/sceneseventschallenge/

Many remaining challenges

- Change expectations of researchers more widely
  - Software and data should be published as well as the data
  - Develop research software knowing it will be published
- Embed software development training into PhD programmes
  - Software Carpentry skills as well as paper writing skills
- Research software should undergo code review
  - Colleagues review software as well as proof-reading paper
- Software and data is the evidence that supports the research
  - Software as supporting evidence should not be “secret”
  - Software IP should not stop research validation and reuse

Many thanks to:

Sound Software and Reproducible Research:

QM: Luis Figueira, Ivan Damnjanovic, Daniele Barchiesi, Steve Welburn, Carl Bussey

Software Carpentry: Greg Wilson (and many helpers)

SSI: Tim Parkinson, Arno Proeme, Neil Chue Hong, etc.

D-CASE Challenge:

QM: Dimitrios Giannoulis, Dan Stowell;

City: Emmanouil Benetos;

IRCAM: Mathias Rossignol, Mathieu Lagrange

Plus: Research collaborators & Advisory Board members

Conclusions

- SoundSoftware funded by EPSRC 2010-2014:
  Support software and data for high quality research in audio & music
- Reproducible Research: Paper + Software + Data. Not Easy!
- Approach: Identify barriers, simple approaches to overcome them:
  - Lack of education / confidence with code -> Training
  - Lack of facilities and tools -> Repository, GUI
  - Lack of incentive for publication

    -> Engagement: Link papers to code, Prizes, Data Challenge
- Major challenges remain – changing views of researchers:
  - Change expectations more widely; Code review;
    Software as supporting evidence for research; IP issues