Pulling everything together

Mike Jackson, ARCHER CSE Team
michaelj@epcc.ed.ac.uk

Reusing this material

Best practices for scientific computing

Wilson G, Aruliah DA, Brown CT, Chue Hong NP, Davis M, et al. (2014) Best Practices for Scientific Computing. PLoS Biol 12(1): e1001745. doi:10.1371/journal.pbio.1001745. http://dx.doi.org/10.1371/journal.pbio.1001745.

1. Write programs for people, not computers
   a) A program should not require its readers to hold more than a handful of facts in memory at once
   b) Make names consistent, distinctive, and meaningful
   c) Make code style and formatting consistent
2. Let the computer do the work
   a) Make the computer repeat tasks
   b) Save recent commands in a file for re-use
   c) Use a build tool to automate workflows

Best practices for scientific computing

3. Make incremental changes
   a) Work in small steps with frequent feedback and course correction
   b) Use a version control system
   c) Put everything that has been created manually in version control
4. Don't repeat yourself (or others)
   a) Every piece of data must have a single authoritative representation in the system
   b) Modularize code rather than copying and pasting
   c) Re-use code instead of rewriting it

Best practices for scientific computing

5. Plan for mistakes
   a) Add assertions to programs to check their operation
   b) Use an off-the-shelf unit testing library
   c) Turn bugs into test cases
   d) Use a symbolic debugger
6. Optimize software only after it works correctly
   a) Use a profiler to identify bottlenecks
   b) Write code in the highest-level language possible

Best practices for scientific computing

7. Document design and purpose, not mechanics
   a) Document interfaces and reasons, not implementations
   b) Refactor code in preference to explaining how it works
   c) Embed the documentation for a piece of software in that software
8. Collaborate
   a) Use pre-merge code reviews
   b) Use pair programming when bringing someone new up to speed and when tackling particularly tricky problems
   c) Use an issue tracking tool

archer | software carpentry | epcc | The University of Edinburgh

Ten simple rules for reproducible computational research

1. For every result, keep track of how it was produced
2. Avoid manual data manipulation steps
3. Archive the exact versions of all external programs used
4. Version control all custom scripts
5. Record all intermediate results, when possible in standardized formats
6. For analyses that include randomness, note underlying random seeds
7. Always store raw data behind plots
8. Generate hierarchical analysis output, allowing layers of increasing detail to be inspected
9. Connect textual statements to underlying results
10. Provide public access to scripts, runs, and results

Sandve GK, Nekrutenko A, Taylor J, Hovig E (2013) Ten Simple Rules for Reproducible Computational Research. PLoS Comput Biol 9(10): e1003285. doi:10.1371/journal.pcbi.1003285. http://dx.doi.org/10.1371/journal.pcbi.1003285.

archer | software carpentry | epcc | The University of Edinburgh

Iterative development

What did you think of this workshop?

- http://www.archer.ac.uk/training/feedback/
- Sticky Notes:
  - One good point on green - learned, enjoyed
  - One bad point on red - confused, bothered
- Optional post-course questionnaire:
  - http://tinyurl.com/2014-12-03-edinburgh-post

Getting access to ARCHER

- Standard research grant
  - Request Technical Assessment using form on ARCHER website
  - Submit completed TA with notional cost in Je-S
  - Apply for time for maximum of 2 years
- ARCHER Resource Allocation Panel (RAP)
  - Request Technical Assessment using form on ARCHER website
  - Submit completed TA with RAP form
  - Every 4 months
- Application for computer time only
  - Instant Access – Pump-Priming Time
  - Request Technical Assessment using form on ARCHER website
  - Submit completed TA with 2 page description of work

Support and Documentation

- Helpdesk
  - Email support@archer.ac.uk
  - via ARCHER SAFE http://www.archer.ac.uk/safe
  - phone: +44 (0)131 650 5000
  - By post, to: Liz Sim
    EPCC, University of Edinburgh
    JCMB, The King's Buildings
    Mayfield Road, EDINBURGH, EH9 3JZ
- http://www.archer.ac.uk/community/techforum/
- http://www.archer.ac.uk/documentation/

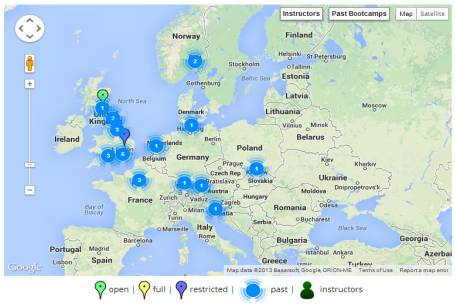
Training opportunities

- ARCHER Training (free to academics)
  - http://www.archer.ac.uk/training/
- Online sessions (using *Blackboard Collaborate*)
  - Technical Forum meetings (normally 15:00 last Wednesday of month)
    - technical presentations of interest to ARCHER users
    - http://www.archer.ac.uk/community/techforum/
  - Virtual tutorials (normally 15:00 second Wednesday of month)
    - opportunity for discussion with ARCHER staff on **any** topic
    - usually starts with a presentation of general interest
    - http://www.archer.ac.uk/training/virtual/
- EPCC MSc in HPC (scholarships available)
  - http://www.epcc.ed.ac.uk/msc/

Funding calls

- Embedded CSE support
  - Through a series of regular calls, Embedded CSE (eCSE) support provides funding to the ARCHER user community to develop software in a sustainable manner for running on ARCHER. Funding will enable the employment of a researcher or code developer to work specifically on the relevant software to enable new features or improve the performance of the code
  - Apply for funding for development effort
  - Planned every 4 months
  - 4th call closes at 4pm on Tuesday 13th January 2015
- See http://www.archer.ac.uk/community/eCSE/ for details

Getting involved in Software Carpentry



admin-uk@software-carpentry.org

http://software-carpentry.org/workshops

http://software-carpentry.org/lessons.html

Software Sustainability Institute

www.software.ac.uk

We want the research community to

- Recognise software as a fundamental research output
- Recognise the value of research software engineers
- Receive better software training
- Recognise the role of software in reproducibility

What we do

- **Fellowship**: we find and nurture researchers with combined talents in research and software
- **Software Carpentry**: teaching basic software skills to researchers from all disciplines
- **Consultancy**: providing software expertise to advance research software
- **Campaigning**: raising the profile of software in research and highlighting the issues it faces

Not broken software broken science but…

We are supported by EPSRC under grant EP/H043160/1

Thank you!

archer | software carpentry | epcc | The University of Edinburgh