soundsoftware.ac.uk
Sustainable Software for Audio and Music Research

Software Carpentry in a Research Group

September 2014 – Published under a Creative Commons CC-BY 3.0 licence

Contents

Motivation

Software development is often a necessity for academics, particularly within scientific research groups. But it is typical for a researcher to have never been taught how to produce, use, or share software efficiently. Alternatively, a researcher may have a background with software, but to be unsure how this can be applied to an academic environment.

Software Carpentry is an organisation of volunteers aiming to teach academics how to use a number of basic software tools and methods, to help them spend less time dealing with software issues and more time doing research.

This document details how to run some basic Software Carpentry or Software Carpentry-inspired learning workshops within a Research Group. It will be useful for any research groups who regularly require software development, particularly in the computational sciences.

What is Software Carpentry?

Software Carpentry (http://software-carpentry.org/) is an organisation of volunteers, now affiliated with the Mozilla Science Lab, teaching academics how to be more productive by teaching them basic software skills.

The team consists of a number of instructors around the globe who present 2-day workshops covering the following topics:

- Unix shell scripting and task automation;
- Version control (typically using Git, Mercurial or Subversion);
- Software development and design (typically using Python or R); and
- Unit testing using an xUnit-style framework.

The following topics may also be covered, depending on the audience:

- Numerical programming using NumPy;
- Regular expressions; and
- Relational databases and SQL.

A full list of previous and current curricula can be found at http://software-carpentry.org/lessons.html.

The aim is not just to teach specific techniques or technologies as to teach standard tools and methods for assembling and running software in a reliable, repeatable, and automatable way.

It is particularly important that the techniques offered should be accessible and relevant to researchers. Software Carpentry aims to achieve this through its learning style, in which the material is worked out "live" using a computer connected to a projector with the learners following along on their own computers, and also by encouraging researchers who have attended these workshops to become instructors for future events rather than relying on professional training staff.

If this sounds relevant to your research group, then we would encourage you to participate in official Software Carpentry workshops in order to pick up training in how to run them and how to communicate the material effectively. We will discuss that below. However, it is also possible to run Software Carpentry-inspired learning events independently within your group; we will follow the discussion with some considerations for those considering running similar events in this way.

First steps

It is strongly recommended that your first step should be to attend a Software Carpentry workshop and consider how to get a Software Carpentry-trained instructor involved in your own workshops.

(To use the Software Carpentry name and logo, you need to get permission to host your event from the Software Carpentry team. Permission is typically granted if the course covers all the core lessons and is taught by at least one Software Carpentry-trained instructor.)

There are two ways to acquire an officially credited instructor for your event: either contact someone externally who can attend it, or allow a volunteer within your group to take the free online training course. See "Instructors" below for details.

Planning an event

Consider who the audience should be: whether it is aimed at both longer-term academics and beginning PhD students, for example. The frequency of the event should depend on how often new academics join your institution. This may be easier to predict with PhD students than other academics. However, the training would be beneficial for all academics, and while there may be a better "return on investment" from training researchers early in their careers, later-stage academics may benefit from being able to encourage good practices in students they supervise.

In many institutions, it's typical to host an annual research skills course for PhD students and new academics. Other institutes have taught research courses for other post-graduate levels (for example Masters students). It may be possible to schedule this workshop within that period. You should then consider if the course should be compulsory for participants.

The workshop could either be private (within the institute) or public (allowing access from other institutions), a decision that will probably depend on how many researchers are in your group. It is usually a good idea to be somewhat "field-specific", that is, to have attendees who are working in a similar subject and ideally a teacher who is familiar with that field.

Consider the timing (and what commitments participants might be involved in, e.g., a large conference?) and the duration of the event. Shorter events may be better-attended, as well as less tiring and more productive, but a certain duration is necessary to get sufficiently deeply into the subject. A single day (or two afternoons) is probably too sketchy. Most Software Carpentry workshops are two days. There have been successful three-day examples, particularly following a pattern of two days of non-subject-specific material and one day applying that material to the research field. We once organised a five-day workshop, and that was definitely too long!

For smaller research groups or those within which the arrival of new researchers is less frequent, it may be more realistic to collaborate with multiple groups within an institution or similar groups within another institution. For larger research groups, it might be useful to collaborate with other groups for core modules, but separately teach the use of more specific software tools more relevant to the group.

For the day, you should organise time for breaks and lunch. You might also want to consider providing refreshments (e.g., tea, coffee and biscuits) and lunch.

Participants

It is not necessary for participants to have any prior programming experience, but it is a very good idea to find out beforehand how much experience they have. It might be useful to separate those with stronger experience into a more advanced group so that everyone would benefit equally, or to ask them to help out other learners on the day. If you have the luxury of enough time to teach the more experienced learners separately and *then* ask them to help out as well, that could work too!

Participants are usually expected to bring their own laptops. This ensures that all the tools and examples are readily available for them after leaving the event. They should be given an installation summary before the event to ensure that they are equipped with all the necessary tools, otherwise this might waste a lot of time on the day. You should also provide contact details for any participants who are struggling to install any dependencies.

Some events have asked participants to download a virtual machine image, with all the tools preinstalled. To use this, a free and openly available tool like VirtualBox will need to be installed on each of the participants' machine. You might also want to consider providing some machines for those who might not be able to bring their own.

Instructors and helpers

A class might contain between 10 and 50 participants. For smaller classes, success might in theory be achieved with one trained instructor; however, an intensive two-day workshop is fatiguing to teach, so you should ensure that at least two people are prepared to teach parts of it, even if only one is formally certified.

Besides the instructor(s), you will need roughly one helper per 10 learners who is prepared to circulate and help out with any technical or hardware problems or misunderstandings. A large factor in the success of these workshops is the preparedness of people to leap in and help out individually when someone gets stuck, and this happens often.

Contacting an External Instructor

There are number of certified instructors who are willing to teach at external institutes. A list of them can be found on the Software Carpentry website (http://software-carpentry.org/pages/team.html). The team should be contacted via admin@software-carpentry.org. As instructors are volunteers, their travel and accommodation should be funded by the host, if applicable.

In the UK, Software Carpentry training is currently coordinated by the Software Sustainability Institute (http://software.ac.uk).

Becoming a Certified Instructor

While it might be feasible to invite an external instructor to the institute for an occasional event, there are benefits for an individual to become a certified instructor. These include:

- bettering your own understanding of software carpentry tools;
- building a reputation within external communities;
- gaining experience in and practicing teaching; and

- helping the science community to progress more efficiently.

To become a certified instructor, a free online training course (http://teaching.software-carpentry.org) can be taken which lasts around 12-14 weeks for around 2-4 hours per week. It isn't necessary to be an expert programmer to take part in the course. In fact, the Software Carpentry team favour instructors who are only a few steps ahead of the students.

After the event

You might consider assigning a coursework-style task to the participants which is delivered after the event. This may be beneficial to test both the quality of the teaching and the understanding of the individuals. Try to resist making the coursework a compulsory assessed exercise: we did this once with poor results (it was too soon for all students to apply the lessons thoroughly, and they became discouraged: the purpose is to encourage learners to continue in a particular direction, not the impossible task of teaching them everything they need to know within two days).

You may wish to issue a certificate to all those who participated.

Some participants on past occasions have expressed that the teachings were beneficial, but they didn't understand how to apply them to their research. You might find it useful to host a follow up day to discuss any consequences and problems in their subsequent work.

Finally, consider how to sustain this effort from one year to the next. Ideally, keen students who have been involved in one workshop as learners could be involved as helpers the next time around and instructors the time after.