Use and utility of ELNs and other tools for reproducible science

Q: Are notebook/labbooks style tools, such as IPython Notebook or Labtrove, useful tools to improve reproducibility?
Q: How are tools such as Github, iPython Notebook and RStudio being used to practice reproducible science?

What are the five most important things learnt during this discussion:

1. Certain tools intermingle code and data. Raises issue of implicit data contexts which inhibits reproducibility since the notebooks can't be rerun ... and may fail (cryptically).
2. Interactive IPython-style notebooks could be used for future teaching as the code and context is inline and integrated. The story of the analysis done. A bridge between reading about what's happened and seeing it, and then experimenting with different parameters etc.
3. Versioning of software can be a challenge as scientists may not be aware of the major impact on reproducibility this can have. And code authors may not be diligent in their versioning policies.
4. Reproducing/replicating the original author's dodgy data counts as reproducibility.
5. Reproducibility "in principle".

What are the problems, and are there solutions?

See above.
Risk of a false sense of security if using the right tools and processes but writing the wrong thing
Being open in processes, tools, versions, data at least provides transparency and the hope that someone will spot a flaw

What further work could be done, and who should do it (make a pledge)?

SSI - Hack-up example of using GitHub to store data and ELN hooked into a continuous integration server. Discuss with Rob Davey.

What software tools or development infrastructures

help researchers achieve reproducible research?

Our discussion has leant towards: repeatability and replication (rather than reproducability, though repeatability and replication are necessary precursors).

What are the five most important things learnt during this discussion:

- Problem: data curation. Will this data be available/readable in decades? Centuries?
- Problem: heterogeneity: tools, languages, libraries, data formats, hardware
- Goal: education and training
- Goal: as much automation as possible, from the start (for example, make is very useful).

Solution:

Workbooks (e.g., Knit-R, IPython) and workflow capturing systems (e.g., virtualisations, Docker)

What further work could be done, and who should do it (make a pledge)?

- When reviewing: asking is code is available?
- Or further, acceptance conditional on data/code availability (to enforce)
- Writing to editors suggesting greater openess and sharing
- A "Checklist" for data/code openness (e.g., data included, Readme to describe build, data, etc.)

Useful resources that people should know about:
Github, Figshare, make, rake, Knit-R, IPython, Docker, Galaxy, virtualEnv (for Python users)

What software tools or development infrastructures help researchers achieve reproducible research?

Five most important things:

- making easy the use of the existing tools, no unreadable file
- creating log for the the gui operation to replicate offline by others
- auto-completion like tool based on AI/statistic
- validation/unit test like for behaviours/ workflows testing

What are the problems, and are there solutions?

Problems:

- Lack of knowledge/confidence /laziness/motivation
- Publishing constraints/short term goals/ short and dirty solutions/ maintenance/support of code
- Lack of awards to provide good sw development

Solution:

- Documentation (lack of knowledge)
- Education/ training
- crediting software development as academic outputs

What further work could be done, and who should do it (make a pledge)?

Smart lab notebook that generate the log of operation/settings to reproduce the experiment using sw tools.
behaviour/workflows validation mechanism like unit tests

Are there any useful resources that people should know about?

Continuous integration, burrito.

What problems should Recomputation.org address?

What are the five most important things learnt during this discussion:

1. We discussed architecture of the envisaged recomputation cloud service and it's a challenging problem
2. It is important to make the submission of reproducible experiments reproducible itself
3. It may be useful to have One-to-many relation between VM core systems and virtual disks.
4. Getting different levels of adopters (alpha, beta, production) with different expectations
5. helping us to refine the development
6. Platform independence

What are the problems, and are there solutions?

- Getting the community of contributors submitting their experiments.
    - tools should make it easy. It should be possible to submit just the experiment data in an
    - archive, not the whole VM. If we can't get it to work, offer login to the submitter.
    - those who want to submit VM, should be also able to do that.
    - not only technical side. The social side lies in evaluating and providing timely feedback.
- Platform independence or catering for multiple VM providers.
    - admin runs automated porting to other providers and sends results for the submitter for approval.

What further work could be done, and who should do it (make a pledge)?

Ongoing project involving St Andrews, SSI and Microsoft Research Cambridge. Follow recomputation.org

Are there any useful resources that people should know about?

- http://recomputation.org/
- http://www.slideshare.net/a_konovalov/spls-recomputation-talkfigshare