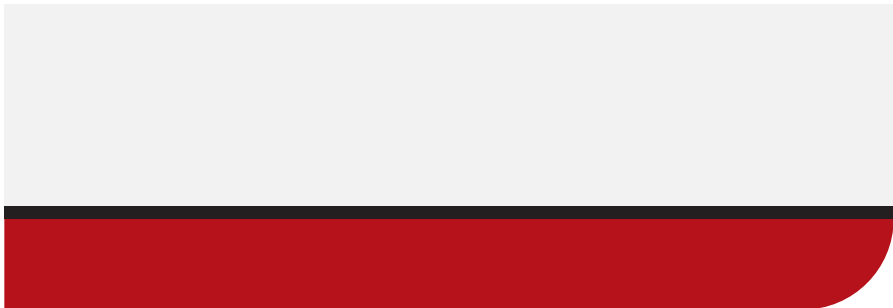


# Sustainable Software

## or “Is your research software correct?”

Robin Long  
September 15, 2017



# Sustainable Software Institute

---

- Software is fundamental to research, 7/10 UK researchers say it is vital to their research.
- Better software leads to better research.
- Better software should be sustainable.
- Sustainable software needs less effort in the long run.

# Is your research (software) correct.

---

- How do you know that your code does what it is meant to?
- We really on code, for science, but science is not always applied to our code.
- Can you reproduce what you did?
- How sure are you of your results?

# We have a problem

---

- A lot of these problems would have been caught with better software, but what does that mean?
- A different approach.
- Be more rigorous.
- Apply principles of testing and reproducibility.

# Are you sure you did that?

---

- How did you generate the results?
- Point, click, click, drag, click, drag, right click, click, drag, click, save.
- Are you sure you clicked the correct cell, or highlighted the correct column?
- How reproducible is a mouse click?

# Are you sure you did that?

---

- Solution: automate.
- Many programs support scripting, use this instead.
- If it doesn't, find another program.
- Don't just automate the analysis, automate the whole chain:  
Data selection, manipulation and analysis.

# Make you life easier

---

- We tend to write programs that are easy for computers to understand, but difficult for humans.
- How do you know it is correct if you struggle to read it?
- Use a high level language.

## Why a high level language

“Programmers write roughly the same number of lines of code per unit time regardless of the language they use”

- Best Practices for Scientific Computing,  
PLOS Biology, Wilson Et Al

# Go to ludicrous speed

---

- What about speed?
- Computing time is (relatively) cheap, researcher time is not.
- High level languages are usually more than fast enough.
- Get the code right first, then worry about speed later (if you even need to).



# I didn't mean that version

---

- myCode\_version1
- myCode\_version2
- myCode\_version2\_broken
- myCode\_version2\_fixed-ish
- myCode\_version2\_fixed
- myCode\_version2\_broken\_again
- myCode\_version2\_fixed2
- myCode\_version2\_fixed\_withChanges

# I didn't mean that version

---

- This is scary.
- Not even clear which the final version is.
- Are you sure of which version went into the paper.
- If the analysis was split, did you all use the same version?
- Version control helps.
- There are many options, if you are starting out from scratch, I recommend git.

# Get a code buddy

---

- Does not even have to understand your research.
- Can tell you where things could be done better.
- Might spot mistakes that you cannot see.
- Problem: Can they even get the code to work on their machine
- Bonus points if it is on a different OS.

# Share your code

---

- Let us assume that you have done all the changes so far:
- You have automated your analysis (one command and you get results).
- You have had someone look over it.
- It is in version control so you know what changes have been made.
- Why not upload it to a public github?

# Share your code, Why?

---

- Science is (or should be) about openness and transparency, we should share.
- Saves other people re-inventing what has already been done.
- People can use your code and cite it.
- Opens the door to more collaboration and **impact**.
- Others can view your code, submit modifications and even enhance your work.

# Are you afraid of your code?

---

- How confident are you about making changes to your code?
- Are you sure it won't break.
- Even if it works do you know the answers are correct?
- This is where testing comes in.
- Most languages have a testing framework you can use.
- Unit, Functional, and Integration tests.

# Are you afraid of your code?

---

- It can seem like a lot of work at the beginning, but it will be worth it.
- Looking back our first examples, testing could have prevented many of these issues.
- Integrate tests with version control.
- Automated testing, you write the tests, they are run periodically to check state of code.

# Problems and solutions

---

- Reproducibility - Learn to script and automate.
- Readability - Write in a high level language.
- Traceability - Use version control.
- Accuracy/Correctness - Testing.
- Reproducibility / Correctness - Share your code / Get a code buddy.



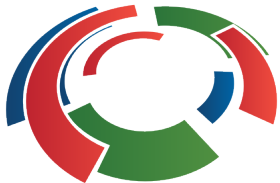
# Is your research software correct

---

Is your research software correct?  
It might be, but can you be sure?

# Help exists

---



Software  
Sustainability  
Institute

[www.software.ac.uk](http://www.software.ac.uk)

# The Software Sustainability Institute

---

The Institute was founded to support the UK's research software community - a community that includes the majority of UK's researchers.

Our mission is to cultivate better, more sustainable, research software to enable world-class research (**better software, better research**). Software is fundamental to research: seven out of ten UK researchers report that their work would be impossible without it.