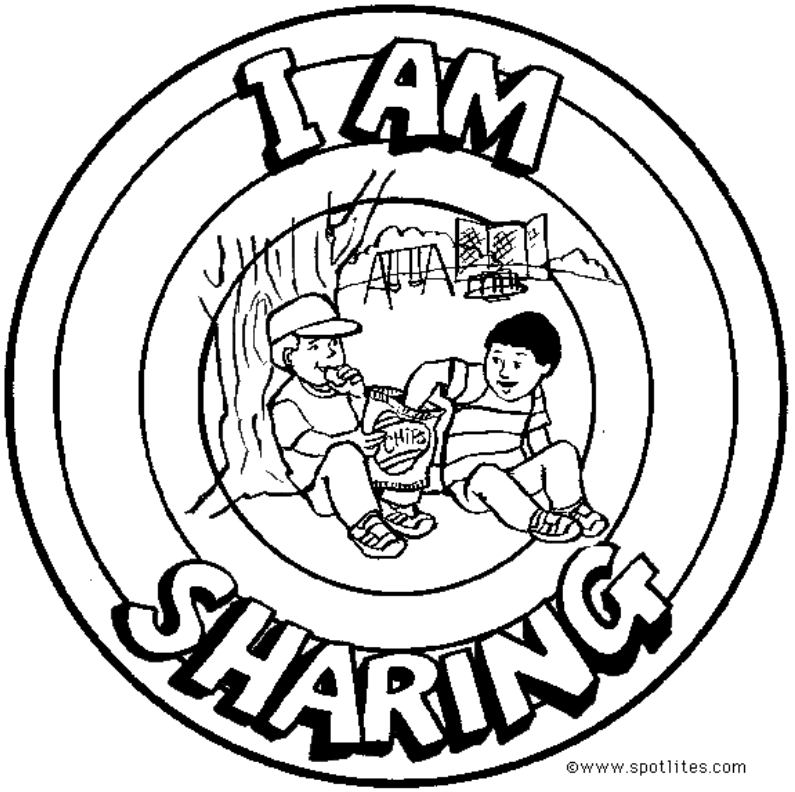
Risks and benefits of an open-source science:
is it possible to make research freely available?



Pietro Berkes

Institute for Theoretical Biology
Humboldt-Universität zu Berlin

Preamble

- Discussion
- I tried to keep arguments practical instead of ideal
- Things I'm *trying* or *would like* to do (i.e., this is not a lesson)

Sharing research

Ideally, scientific data, methods, and analysis tools should be shared with the scientific community in order to let others

- verify
- reproduce
- extend

the results.

Printed resources

- This is in principle what papers are for!
- But we all know:
  - the description of data and results is incomplete (usually given through statistics)
  - the description of the methods is in many situations reported in natural language and is approximative
- How often after reading a paper we are left with some doubts about the details?

Additional resources

- „It must (...) be possible from any published figure or table to find the raw data which form the basis of the results. The advantage of such a system is also that it will be easier to find unpublished data if at a later time they may perhaps be easier to interpret or are suitable to use in a new context. “

“Guidelines for good scientific practice”,
DCSD, 1998

Online resources

- Particularly relevant for theoretical neuroscience
- Easy: input data and source code can be published online
  - the source code can be verified
  - it provides an explicit description of the algorithms (e.g., TDSEP)
  - easily extendible (e.g., to explore the parameter space)

Why?

- Algorithms and data available:
  - to try out an idea or to make a comparison with other algorithms is a matter of hours
- Not available:
  - to implement and debug an algorithm is a matter of days
- (The problem is not only to publish the source code but also to make it reusable)

Why not?

- Requires additional efforts
  - clean code
  - documentation
  - HTML pages
  - archive and maintain the code
  - important for the community to agree on a set of practices (for rapid but reliable development), conventions (for sharing), and tools (for development and publishing)

(continue)

Why not?

- Makes things simple for colleagues
  - credits and citations (e.g., van Hateren's Natural Stimuli Collection, MNIST database, ...)
  - “homework” effect
  - it makes things simple for you, too! (if you're not the only one sharing, of course)
- Even a small error in a computer simulation invalidates the results

Open questions

- How to make results really reproducible?
- How to prevent errors?
- How to react when errors are found?
- Where to publish the data?

Reproducibility

What should be published to ensure complete reproducibility?

- Input data
  - neutral format (plain text or binary, XML)
  - format description or simply separate functions that load the data in meaningful variables
  - date of entry and the identity of the person(s) responsible for carrying out the experiment (DCSD, 1998)

(continue)

Reproducibility

- Source code
  - the code used to perform the simulations, not a cleaned-up version
- Results
  - they could in principle be generated
  - useful if the processing time is very long
  - useful if the hardware or software got out of date

(continue)

Reproducibility

- List of all external factors: platform and operating system, libraries, compilers, interpreters with their exact version number
  - users with old software
  - upgrade in the libraries that might make results better, but not reproducible
  - not very useful with proprietary software

(continue)

Reproducibility

- possible aids:
  - CVS
  - platform-independent languages (although numerical computations are rarely platform-independent)
  - backward-compatible changes
- Random seeds and constants
  - no hard-coded constants (they form a list of all ad-hoc parameters)
- Did I forget anything?

Avoiding errors

- Each result (especially if important) should be independently reproduced at least internally (DFG, 1999)
- Not very realistic. Alternatives:
  - unit testing:
    - write code in “minimal testable modules”
    - automation of testing is crucial (JUnit, PyUnit, CPPUnit, ...)
    - encourages writing cleaner code
    - works as a guaranty for users

(continue)

Avoiding errors

- Alternatives (2):
  - XP: pair programming
  - use widely used basic routines (LAPACK, Matlab, ...)
- (Use a high-level programming language)
  - speed issue: often not an issue (e.g., Scipy uses LAPACK libraries just as Matlab does)

Avoiding errors

- Teaching basic knowledge:
  - internals (e.g., floating-point representation)
    Area of a triangle:

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

If a = 9.0, b = c = 4.53 and precision = 2 digits:

correct values: s = 9.03, A = 2.34

found: s = 9.05, A = 3.04

New formula, stable also for flat triangles:

$$A = \frac{[(a+(b+c))(c-(a-b))\cdot(c+(a-b))(a+(b-c))]^{1/2}}{4}$$

(continue)

(Goldberg, 1991)

Avoiding errors

- Teaching basic knowledge (2):
  - numerical algorithms
  - tools mentioned before

Reacting to errors

- Someone reading or using the code does find a bug, despite all preventive steps. How to react?
  - If the results were structured and intuitive, probably the bug won‘t change them qualitatively.
  - Of course, for important bugs one has to post an *errata*. But were is the limit?
  - For tiny errors, if the code is published in an official and stable repository one could post a patch and the new quantitative results there.

Repository

- Where can one publish its resources?
  - stable: high mobility of researchers, links on publications
    - DFG, DCSD, ...: data should be stored for at least 10 years
  - original data should be stored at the institution which produced them and the individual participants in the research projects can take copies of the data with them (DCSD, 1998)
  - current repositories perfect to store software libraries (e.g., Sourceforge), but not data or single simulations
  - pilot project: VISIOME, japanese neuroscience portal with focus on the visual system. Repository of data, reprints, models, etc. They even plan to have historical hardware in order to be able to run old software remotely!

Conclusions

- Need for a conscious effort to define a set of rules and standards for the sharing of resources
- Need to begin sharing