
Mark Schmidt
University of British Columbia
What code would you release?

• Implementation of method proposed in your paper?

• Implementation of competing methods?

• Implementation of basic method in your field?

• Experiment scripts?

• “Research” implementation vs. “professional” implementation.
Why would you release code?

• Let others use your method?
• Let others replicate your experiment?
• Help others with their research?
• Publicize yourself?
• Push people to work on other problems?
• Get hard-to-publish implementation tricks out into the world.

Want people to make interface.

Usage outside of your field/academia.

Other people can find bugs.
Why would you NOT release code?

• You plan to make money off of the code?

• You have something to hide?
  - Not enough resources to support.

• You don’t want others replicating your results?
  - Bad at programming. (Buys)

• You don’t want to appear “2nd place” in other papers?
  - Email spam.

• You want people to suffer the way you suffered?
Design Principles

• “Black box” function:
  – One function that does everything.
  – Good if think users don’t care how it works.
  – Check default parameters on many problems.
    • Non-default parameters exposed through arguments.
Design Principles

• “Modular parts”:
  – Many simple with common interfaces.
  – Think carefully about data structures.
  – Good if users need to know how it works.
  – More likely to get user contributions.
Demos

• Always include a simple demo:
  – Something user can run quickly.
  – Gives obvious visual feedback that code is working.

• Good demos can also replace documentation.
Performance vs. Generality

• Trade-off:
  – Spending time optimizing code for specific problems.
    • Vectorization, multi-core, BLAS, etc.
  – Spending time making code apply to many problems.
Programming Languages

• Many potential users don’t know low-level languages.
  – Matlab/Python interface, even if just to .exe file.
  – High-level languages are compatible across more architectures.

• But high-level languages are slow:
  – Use mex/Cython for slow parts of code.
  – But include ‘slow’ version and compile instructions.
Licensing Options

• Completely-free (FreeBSD):
  – Most publicity, least incomes.

• Non-commercial:
  – Similar level of publicity/income to above.
  – May restrict some important applications.

• License code:

• License binary:
User Interface/Support

• Some users really like a graphical user interface (GUI).
  – Can be great for non-power users.
  – Can help visualize what is going on.

• But power users still want command-line access:
  – You will annoy them without this.

• Provide answers to common problems, e-mail for questions.
The REAL objective function

• Will this code solve my problem?
  1. Google search for keywords.
  2. Go to webpage of code, read description.
  3. Check programming language of code.
  4. Download code.
  5. Unzip and compile.
  6. Try out demo.
  7. Try to formulate my problem in language of code.
  9. Go back to 6 if problem is not solved (if multiple times, goto 2 or 1).
  10. Done.

• You can lose a huge number users on any step:
  – Your objective is optimize time to do 1-9, NOT the time to do 8.
The REAL objective function

• Optimizing the real objective:
  1. Give code a meaningful name, place it in a Google friendly location.
  2. Webpage should start with high-level description of what method does.
  3. Don’t use obscure programming languages (OCAML).
  4. Don’t require accepting license or registering e-mail to download.
  5. Make a simple “Makefile”: no manually setting paths required.
  6. Give a simple demo: don’t assume they will read everything.
  7. Make it easy to input new problems to code.
  8. Use a state-of-the-art-ish method.
  9. Give users a list of “common problems”.

Git