Bioinformatics for programmers

Scientific software development: best practices and approaches

Konstantin Okonechnikov
Max Planck Institute For Infection Biology

Летняя школа биоинформатики
Москва, 2013

Objectives

- Get introduced to software development in context of scientific research
- Learn best practices by example

References:

- ***Aruliah, D. A., et al. "Best practices for scientific computing." arXiv preprint arXiv:1210.0530 (2012)***
- ***Prlić, A., & Procter, J. B. (2012). Ten simple rules for the open development of scientific software. PLoS computational biology, 8(12), e1002802.***

Why bother?

- Software is just another experiment technique: must be clean, reliable and reproducible
- Software is reused by others
- Software is scientific result
- Good infrastructure doesn't move you forward, but allows to do it

Write programs for people

- Other people should be able to read your code. Including future yourself.
- Bad examples
  - Perl, but can be any other language:

```
m[1]-- ? m[0]*=2 , f(): printf(m);
```

- Good example
  - Python
- Practices:

  - Clear and **consistent** coding
  - Use classes and objects

Do not reinvent the wheel

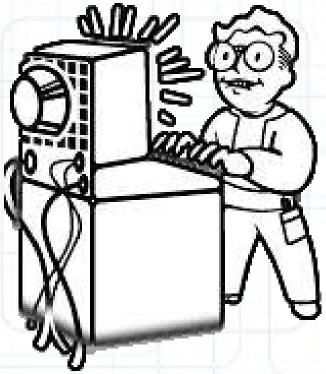
- It was all done before.
- Bad example:
  - Write your own findStr() function
- Practices:
  - Use libraries and existing tools
- **Exception**: reproduce solution to get better understanding of it

Do not copy-paste

- Code should be modularized. Every piece of data should be unique.
- Bad example:
  - 20 similar functions with only one parameter changed
- Practices:
  - Modularize code into functions and libraries.
  - Side effect: code is easy to maintain, easy to test

Automate repetitive tasks

- Rely on the computer to repeat tasks
- Use workflow management systems
- Use build tools to automate the development:
  - It should be possible to build and deploy your project with only one command
- Use IDEs:
  - Easy refactoring
  - Many tasks are available by default



Use version control

- Keep track of the changes in a repository (svn, git, etc.)
- Use existing code hosting platform:
  - Github, bitbucket, etc.
- Everything that has been created manually should be put in version control:
  - Test data
  - Configurations

Make incremental changes

- Work with frequent feedback and course correction
- Good example:
  - Agile software development techniques (Scrum etc.)
- “Release early, release often” (L. Torvalds)

Plan for mistakes

- Every program has bugs. Verifying and maintaining code required time and effort.
- Use assertion to check for operations

```
void f(Data* p ) {
        // p can be not null!
        assert(p);
        processData(p);
        return p;
}
```

Test systematically

- Automated testing
- Good examples:
  - Use unit test frameworks for your language
  - Calculate test coverage
  - Use test data as your sample data later
  - Create test cases for bugs
- Crazy example:
  - Write tests before writing code

Keep It Simple Stupid

- Rule of first launch: your software must be easy to install and use
- Use standard data formats
- Use native distribution methods
- Be your own user
- Documentation helps

Optimization

- Only optimize the code that is working properly
- Find the bottleneck first (!)
    - Use profilers
- Choose a better algorithm
- Analyze your typical input data, than use caching
- Use parallel computing technologies (multicore, cluster, cloud, GPUs)

Develop open-source

- Transparent development promotes scientific progress
- Collaboration is more fruitful than competition
- Release software prior to publication

Build a community

- Know your users and communicate with them
- Make it easier for others to contribute ideas and feedback
- Promote your project: social networks, conferences

Programming exercise

- Learn how to use the practices on an example of demonstration SNP-calling pipeline
- Unpack ***programming_problem.zip***
- Open the PDF file

Some more references

- A. Hunt, D. Thomas “The Pragmatic programmer”
- B. Kernighan, R.Pike “The practice of programming”
- Optimization:

McDonald, E., & Brown, C. T. (2013). khmer: Working with Big Data in Bioinformatics. arXiv preprint arXiv:1303.2223.

Спасибо за внимание!
Thank you for the attention!