Assignment 3 David Grommisch

**Scientific computing at a glance – a literature summary**

As well as wet lab biologists develop their projects researchers in scientific computing need to follow certain best practice directives. The recommended articles provide insights and advice on best practice guidelines for scientific computing and represent a useful resource for researchers that want to transition into various types of ‘computational scientists’ or need to tackle both – wet lab and computational biology – within their projects.

The presented articles share a great deal of advice which will be outlined in the following paragraph.

First and foremost, all authors strengthen the importance of reproducibility not only regarding structure and organization of a project but also referring to documentation, code, and archiving obtained data. While all the authors present different angles on scientific computational biology the main concept of reproducibility of results is there common denominator and implies many other of the mentioned best practice guidelines.

Second, they strongly outline the importance of transparency. Transparency is especially important in computational biology and implies not only sharing a final developed software but even more giving public access to developed code, original data, and the thought process behind a finished project. This can go as far as open development of a whole software.

Finally, all of the authors emphasise the benefits of version control software. It is considered invaluable for the management of developing a computational experiment and additionally provides backup, a history record, and provides the perfect platform for collaborative projects allowing simultaneous development of a project on several layers including documentation, coding, and manuscript writing.

All the above-mentioned points are crucial components of every presented article. However, each of the articles has its very own focus or core theme that will be discussed in the following paragraph.

Starting with Noble, the importance of a good organized project becomes apparent. Giving advice on how to structure a new project he underlines the importance of recording operations, rigorous commenting of your code, and avoiding manual editing of intermediate files in order to develop an easy understandable and therefore good program. Moreover, he gives his insights on the importance of a driver script, using relative pathnames in order to make the program work for colleagues, and finally to create a restartable script. Additionally, he comments on useful habits to develop while coding namely developing robust code by implementing assertions and aborting programs once an error occurs. His final take home message underlines the importance of efficient logistics to produce reproducible experiments.

The second article gives more specific input on creating reproducible computational research. Sandve et al. state that it becomes increasingly important to provide clear protocols, be transparent, and include all data in order to present a good platform for reproducible research. Giving ten rules they aim to give a guideline comprising minimal efforts to develop a reproducible experiment. Those ten rules cover the mentioned topics of version control to be able to backtrack code, recording and documenting intermediate results and parameter choices, and archiving external software used. Furthermore, they include sections about keeping track of the analysis workflow, mention the importance of noting randomness factors, and avoiding manual data manipulation. All in all, this article mentions ten concepts to follow in order to design reproducible experiments and gives specific tools to do so.

While the first and second article are more focused on informing the reader about how to organize a project and how to design reproducible experiments and code the third article specifically mentions resources to learn coding and touches upon a general state of mind a computer scientist needs. Although it states how important version control, reproducibility, and transparency are most of the message is directed to the reader in acquiring a general understanding of algorithms, believing in yourself, and first of all developing functional code (one can write it more elegant afterwards). Loman and Watson supply the reader with clues and resources in order to point the general expectations towards a computational biologist out.

Wilson et al. elaborate in depth on a few best practices for biological computing in general. Most of their points are set up to ensure reproducible, transparent, and version controlled code. Unlike the others, they try to give a little more insight into actual code structure by giving a few specific examples and explaining concepts like defensive programming using assertions, making use of unit testing, and utilizing bugs to test specific cases for a program. Furthermore, they give clear hints on how to best document the experimental protocol and give a first clue on the benefits of collaborations.

Lastly, Prlic and Procter share their view on the benefits of open developments. To give a basis they state the needs they see for good open and collaborative development starting from coding according to certain standards, developing transparent and simple experiments and code, and using already published software or packages. Especially different from the other authors are their insights on growing a community to promote, fund, and sustain a developed software.

Altogether, the authors share their views and good computational practices which includes effective but not so much efficient programming, and creating reproducible, transparent, version controlled, and archived computational experiments. Another important point is publically published datasets that are able to grow a community and make research trustworthy and of common interest. Combined, they give a very neat description of how to tackle the task of a computational experiment, clarify expectations and needs, and supply the right resources in order to follow their advices.

**References:**

1. Noble, A Quick Guide to Organizing Computational Biology, Plos Computational Biology, 2009

2. Sandve et al., Ten Simple Rules for Reproducible Computational Research, *Plos Computational Biology*, 2013

3. Loman & Watson, So you want to be a computational biologist?, *Nature biotechnology*, 2013

4. Wilson et al., Best Practices for Scientific Computing, *Plos Biology*, 2014

5. Ten Simple Rules for the Open Development of Scientific Software, *Plos Computational Biology*, 2012