**Dieudonné Nkulikiyimfura**

**Applied Bioinformatics: Literature on bioinformatics software**

**Question 1**

1. **Paper#1: A quick guide to organizing computational biology projects by William Stafford Noble**

Like any construction of house requires foremost grading and site preparation, working on biology project requires a well structured or organized files and directories as well as documentation of the project progress. These logistics are for an efficient and accurate performance, reproducible computation experiments for anyone and/or your future self. The paper did not deal with good coding practices, methods for automation of experiments, and writing a manuscript based on experimental results.

1. **Paper#2: Ten simple rules for reproducible computational research by Geir K.S. et al.**

A bioinformatician needs to develop good habits of reproducibility in order to ensure realistic and efficient research processes. These include not only reproducing the results but also keeping track of raw data, versions of scripts and used software, documenting all processes but also providing public access to scripts, runs, and results. These habits lower the cost of future maintenance of the codes.

1. **Paper#3: So you want to be a computational biologist? By Nick L. and Mick W.**

What does it take to be a computational biologist? Knowledge of biology is a prerequisite requirement. This is necessary in setting traps to test (analogy to positive and negative controls in wet lab experiment) the reliability of your script or code and in interpreting the computational result. Choosing an algorithm that best suits your data and documenting all processes is a must as well as keeping track of changes made on codes. When you ran into problems, ask for help from experts in the field or from online forum. Be creative and keep learning!

1. **Paper#4: Best practices for scientific computing by Grey Wilson et al**

So now you know your research question and you can build and/or use existing software to answer your question. How efficiently can you be? Are your codes/software reliable?

Write human readable programs by limiting a single task to one function or simply adopt divide-conquer algorithm design paradigm. Let the computer do the work for you and this way you will avoid errors induced by manual manipulations in tackling repeating tasks. Use of version control system to keep track of changes made on codes and/or data. Make a module for reusable codes/data in order not to repeat yourself. Implement defensive programming/automated testing by planning before hand for mistakes. Document everything and ask someone to review your codes for possible bugs.

1. **Paper#5: Ten simple rules for the open development of scientific software by Andreas P. et al.**

You have built robust software to answer your question and you even have published a sound paper out of it! Also, you have published your source code so that others can reproduce your result and use this for their own data. For your software to remain useful after its publication, it needs to be released under an open license. By doing so, other developers will help in sustaining the source code and it is your responsibility to be kind enough in acknowledging everyone’s contributions for this community to grow.

*As conclusion, above authors agree on writing readable, version-controlled, and tested codes for efficient production and reproducibility of the results.*

**Question 2**

1. **Paper#6: Ten recommendations for creating usable bioinformatics command line software by Torsten Seemann**

The paper highlights the necessity of developing user-friendly bioinformatics tool. Although, you need to market your tool as mentioned in paper#5 but an important work needs to be done during the implementation of the tool. Your tool needs to be easy to use or provide hints, feedback, and help to users.

**Evaluation of BWA**

* **If no parameters are supplied, BWA prints software version and usage information**

[dodos07@milou2 ~]$ bwa

Program: bwa (alignment via Burrows-Wheeler transformation)

Version: 0.7.15-r1140

Contact: Heng Li <lh3@sanger.ac.uk>

Usage: bwa <command> [options]

Command: index index sequences in the FASTA format

mem BWA-MEM algorithm

fastmap identify super-maximal exact matches

pemerge merge overlapping paired ends (EXPERIMENTAL)

aln gapped/ungapped alignment

samse generate alignment (single ended)

sampe generate alignment (paired ended)

bwasw BWA-SW for long queries

shm manage indices in shared memory

fa2pac convert FASTA to PAC format

pac2bwt generate BWT from PAC

pac2bwtgen alternative algorithm for generating BWT

bwtupdate update .bwt to the new format

bwt2sa generate SA from BWT and Occ

Note: To use BWA, you need to first index the genome with `bwa index'.

There are three alignment algorithms in BWA: `mem', `bwasw', and

`aln/samse/sampe'. If you are not sure which to use, try `bwa mem'

first. Please `man ./bwa.1' for the manual.

* **BWA does not have a “-h” or “-help” switch, you can instead get manual information by typing**

man bwa

* **BWA doesnot support “-v” or “-version” switch**
* **BWA use stdout for message and errors and if something goes wrong, it prints the usage information**

[dodos07@milou2 ~]$ bwa mem reads.fastq

Usage: bwa mem [options] <idxbase> <in1.fq> [in2.fq]

* **BWA fails to validate parameters**

bwa index -b N reads.fastq

[bwa\_idx\_build] fail to open file 'reads.fastq' : No such file or directory