Week 7, Lecture 13

István Albert
Bioinformatics Consulting Center
Penn State

Instrumentation <-> Methods

- Sanger instrumentation → few long reads → finding similarities between genomes → local alignment → homology

- High throughput sequencing → very large number of very short reads → semi-global alignment → matching known information

Processing sequencing reads

Sequence DNA fragments (DNA library)

Unknown Genome → De novo assembly
Know Genome → Read mapping (alignments)
Known genome (reference)

Gene discovery (annotation)
chip-Seq, RNA-Seq, SNP calling etc.

Alignment concepts

<table>
<thead>
<tr>
<th>perfect match</th>
<th>one mismatch</th>
<th>insertion vs ref.</th>
<th>deletion vs ref</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCAAG</td>
<td>GCAAG</td>
<td>GCAAG</td>
<td>GCAA-G</td>
</tr>
</tbody>
</table>

**AGCAAGTATGTAAGGGCGAGAAAGGCAAG**

NOTE: mismatches or indels can be longer than 1 base!
It gets complicated very quickly
Alignment scoring depends on mismatch scoring (different across bases!), gap open, gap extension penalties
Challenges: repetitive and low complexity regions – where to place the query?

Challenges: genomic reorganization can easily lead to mis-mapping

Reference genome (aka the illusion)

Real genome (with small rearrangement)

This is the query

Mismapping against the reference

Short Read Mappers (aligners)

• Use **heuristics** to quickly identify locations (hits) where the reads match

• Heuristics → not all hits will be found!

• Tradeoff: **resource usage** vs **speed** vs **accuracy** vs **usability**

• Each domain of application may have more appropriate tools

A few popular aligners

• BWA, bowtie, SOAP2 and many others

There is no single best tool, the issues to consider:

1. **documentation** → can we figure out how it works
2. **input features** → what type of input can it handle
3. **reporting features** → will it produce the type of output that we can use
4. **performance** → is it feasible to run on my resources
Aligner features

• Some cannot handle indels (insertion/deletions) → these tools are typically extremely fast will run on a laptop for even large genomes (bowtie 1) → application Chip-Seq

• May or may not use the quality score during the alignment (usually not, that would alter the alignment score)

• Differences in the they report alignments:
  – unique alignments only (note that this is an ill defined concept!)
  – best alignments above a cutoff
  – all possible alignments (can slow down the process greatly)

Tool comparisons

The vast majority of tools comparisons lead to “red-herring” type fallacies

1. Tool A is faster than Tool B and we tacitly assume that the results are the same

2. Tool A is more ‘accurate’ than Tool B and we assume the resource utilization is be the same

Robust results should be tool independent!

• We need tools that are correct and finish running within our resources

• Results should be hold across methods

• There may be exceptions to this – but those need to be explicitly documented
Tool installation

• A bioinformatician’s job requires to evaluate and install a large number of tools

• The ease of installation usually correlates with the quality of tool

• Documentation is essential → otherwise it is no more than a black box

Steps to installing tools

Determine the distribution type

1. Executable (binary) code.
   Download the code and you are done. Easy to install → but it may not be optimized to your system.
   It could also contain undesired functionality.

2. Source (text) code.
   Download the code and compile it

Determine the type of the source code

1. Source is of a compiled language that will be turned into a binary program (typically C but could be others)

2. Source is of an interpreted language that will run the code: java, perl, python, ruby
Check list for source code that needs compilation

1. Does it have a configure script? If yes then run it
   
   ./configure

2. Now run make
   
   make

Ideally you should be done. This will create the binary.

(A program may need library dependencies. Then those need to be installed with the package manager)

Checklist for interpreted languages

1. You need to have the language installed. Most modern computers have perl, python, java installed by default.

2. The source code may have “dependencies” – a much dreaded word could lead to a lengthy procedure of downloading other code that in turn may depend on other and other etc...

Automated installation

- Language specific – will require installing a language specific package manager

- Python has easy_install and pip, Perl has MCPAN, ruby has gem

  easy_install install package-name

  or

  pip install package-name

Installing good tools is very easy – not so good ones are mini puzzles – badly designed tools are incredibly frustrating

BWA (Burrows-Wheeler Aligner) by Heng Li

Burrows-Wheeler Aligner (BWA) is an efficient program that aligns relatively short nucleotide sequences against a long reference sequence such as the human genome. It implements two algorithms, bwa-short and BWA-SW. The former works for query sequences shorter than 200bp and the latter for longer sequences up to around 1000bp. Both algorithms do gapped alignment. They are usually more accurate and faster on queries with low error rates. Please see the BWA manual page for more information.

http://bio-bwa.sourceforge.net/

Download, unpack, compile with make and link to bin

Read also the bwa-mem controversy – the bwa-mem paper rejection
Download – unpack and make

Uses `bz2` compression hence the `j` flag (the `z` command would expand a gzip file)

Self discoverable tools

Aligning reads with BWA-MEM

For BWA is three step process (this is different for each aligner):

1. **Index the genome** – this only needs to be done once for each genome → `bwa index` ...

2. **Create the alignment** → `bwa mem` ...

Homework 14

Align a query file with both BWA and BLAST and look at the output of both processes.

Both are sequence aligners, what seems to be main difference that you observe? Which one seems to run faster?