Week 11, Lecture 21

István Albert

Biochemistry and Molecular Biology
and Bioinformatics Consulting Center

Penn State
Topics to be covered in the next lectures

1. Assembly
2. Metagenomics
3. Chip-Seq
4. RNA-Seq
Genome Assembly

The path to a whole genome

Multistep process

1. **Assemble** short reads into longer sequences → contigs

2. **Scaffolding** – arrange contigs relative to one another based on external information

3. Genome **finishing** (gap closure) → fill in gaps with directed sequencing procedures and manual curation
AMOS – A Modular Open-Source whole genome assembler

It is not a single software rather than a collection of interoperable tools, standards and techniques.
Read assembly challenges

Repeated elements: RPT A1 and RPT A2

A valid assembly of two contigs instead of one

Other challenges:

- genomic variation, heterozygosity, copy number variation
- misassembly due to sequencing errors
- chimeric

Images and content from
http://www.cbcb.umd.edu/research/assembly_primer.shtml
Scaffolding

• Orienting contigs via paired end (or mate-pair) information

Additional information to assist the process:

• use alignment positions in related genomes
• use gene synteny (co-localization of genetic loci)

There are fewer automated pipelines:

**BAMBUS** – Hierarchical Scaffolding With Bambus by M. Pop, D. Kosack and S. Salzberg
Finishing genomes

Correspondence

Finishing genomes with limited resources: lessons from an ensemble of microbial genomes

Niranjan Nagarajan, Christopher Cook, MariaPia Di Bonaventura, Hong Ge, Allen Richards, Kimberly A Bishop-Lilly, Robert DeSalle, Timothy D Read, and Mihai Pop

1 Computational and Mathematical Biology, Genome Institute of Singapore 127726, Singapore
2 Center for Bioinformatics and Computational Biology, University of Maryland, College Park, MD 20742, USA
3 Naval Medical Research Center, 503 Robert Grant Avenue, Silver Spring, MD 20910, USA
4 Henry M. Jackson Foundation, 1401 Rockville Pike, Rockville, MD 20852, USA

BMC Genomics
Volume 11

Finishing Toolbox

The process of finishing a genome and moving it from a draft stage (the result of sequencing and initial assembly) to a complete genome is typically a time and resource intensive task. The advent of new sequencing technologies has come with its own set of opportunities and pitfalls in the finishing process. While genomes can now be sequenced to high redundancy in a cost-effective manner, the process of assembling the genomes is more challenging and often draft genomes are fragmented into hundreds of contigs. Correspondingly, the task of producing the complete genome can involve months of lab work and thousands of finishing experiments and is usually done in large genome centers.

The work in our lab has focused on computational approaches to speed-up the finishing process. Specifically, we have explored the use of optical mapping and mate-pair data to augment assemblies and direct finishing experiments. The tools developed in our lab have been used in several finishing projects, producing complete genomes (and near-complete ones) with surprisingly little computational and experimental effort (Nagarajan et al., in submission). The executables (as well as source code) for these tools are freely available here:

- Scaffolding using Optical Restriction Mapping
  Optical Maps are global, ordered maps of restriction site locations in a genome. This information can be quite useful in scaffolding contigs from a shotgun assembly to guide the finishing process. A set of programs to exploit optical maps for assembly can be found here: SOMA v2.0 (63 MB tar.gz file). This version of SOMA contains several improvements to programs in v1.0 as well as new scripts for working with multiple maps, contig graphs and scaffolds.
Genome assembly is an art

Many different approaches – substantial supervision/evaluation required at each step of the process.

Genomes can vary greatly in complexity – genome size/repetitiveness is usually the limiting factor

Constant tuning and evaluation is needed.
The N50 statistic

- N50 length is defined as the contig length L for which 50% of all bases in the sequences are in contigs of length less than L.

1. Sort all contigs by size from highest to lowest

2. Compute cumulative sum of lengths

3. Smallest number of contigs that add up to the half of the assembled length
Using the Velvet Assembler

Velvet
Sequence assembler for very short reads

- Current version: 1.2.08
- Manual and extension for Columbus in pdf format
- Public Git URL: git clone git://github.com/dzerbino/velvet.git
- For up-to-date info, you can consult and/or subscribe to the mailing list.
- For transcriptomic assembly Velvet is extended by Oases.

Download, unpack, and make Velvet

Download the 23.tar.gz dataset from the webpage

Velvet Assembly is a two step process:

• velveth $\rightarrow$ builds the hashtable
• velvetg $\rightarrow$ run the from the hashtable
Running velvet single end

```
# single end assembly

# generate the test dataset
wgsim -N 100000 refs/mini.fa r1.fq r2.fq > mutations.txt

# create a single end file
cat r1.fq r2.fq > single.fq

# create a hash of size 31 and store
# the results in the directory called s31
~/bin/velveth s31 31 -fastq single.fq

# assemble the reads
~/bin/velvetg s31 -exp_cov auto

[7.514575] Estimated Coverage = 4.353741
[7.514587] Estimated Coverage cutoff = 2.176871
Final graph has 2304 nodes and n50 of 667, max 3464, total 914609, using 166209/200000 reads
```
# paired end assembly

# find and link the shuffleSequence_fastq.pl file
# see velvet/contrib
~/.bin/shuffleSequences_fastq.pl r1.fq r2.fq paired.fq

# create a hash of size 31 and store
# the results in the directory called s31
~/.bin/velveth p31 31 -fastq -shortPaired paired.fq

# assemble the reads
~/.bin/velvetg p31 -exp_cov auto

[7.723953] Writing into graph file p31/LastGraph...
[7.911533] Estimated Coverage = 4.340741
[7.911544] Estimated Coverage cutoff = 2.170370
Final graph has 333 nodes and n50 of 49270, max 84930, total 1059762, using 166578/200000 reads

ialbert@porthos ~/.work/lec21
A few observations

• Paired end assembly leads to radically better assembly $\Rightarrow$ N50 of 667 vs 49270

• Hash size matters $\Rightarrow$ how to pick the right one? Experts say to try “explore” the parameters.

• VelvetOptimizer.pl (in the velvet contrib)
Other resources: quite a few review papers

• A Practical Comparison of De Novo Genome Assembly Software Tools for Next-Generation Sequencing Technologies (PLoS ONE 2011)

• GAGE: A critical evaluation of genome assemblies and assembly algorithms (Genome Research, 2011)
Exploring single-sample SNP and INDEL calling with whole-genome de novo assembly

Heng Li

Received April 3, 2012.
Revision received May 2, 2012.
Accepted May 3, 2012.

Bioinformatics (2012) 28 (14):

Abstract

Motivation: Eugene Myers in his string graph paper suggested that in a string graph or equivalently a unitig graph, any path spells a valid assembly. As a string/unitig graph also encodes every valid assembly of reads, such a graph, provided that it can be constructed correctly, is in fact a lossless representation of reads. In principle, every analysis based on whole-genome shotgun sequencing (WGS) data, such as SNP and insertion/deletion (INDEL) calling, can also be achieved with unitigs.
Assembly evaluation

• Often feels surprisingly ad-hoc (people write home grown scripts to fetch statistics/subselect contigs etc)

• AMOS – contains visualizers hawkeye

• To compare to related genomes we need optimal aligners not short read mappers!
Aligning contigs

AMUMMERA3BL
MUMMER 3+
TMUMMER 3DR

Ultra-fast alignment of large-scale DNA and protein sequences

download and install

Overview

MUMmer is a system for rapidly aligning entire genomes, whether in complete or draft form. For example, MUMmer 3.0 can find all 20-basepair or longer exact matches between a pair of 5-megabase genomes in 13.7 seconds, using 78 MB of memory, on a 2.4 GHz Linux desktop computer. MUMmer can also align incomplete genomes; it can easily handle the 100s or 1000s of contigs from a shotgun sequencing project, and will align them to another set of contigs or a genome using the NUCmer program included with the system. If the species are too divergent for a DNA sequence alignment to detect similarity, then the PROmer program can generate alignments based upon the six-frame translations of both input sequences. The original MUMmer system, version 1.0, is described in our 1999 Nucleic Acids Research paper. Version 2.1 appeared a few years later and is described in our 2002 Nucleic Acids Research paper, while MUMmer 3.0 was recently described in our 2004 Genome Biology paper. We have also developed a GPU accelerated version of MUMmer called MUMmerGPU.

For more information regarding the MUMmer package, please refer to the:

Distribution
Online manual
Online tutorials

To receive software update notices, please join the users' mailing list. This list will only be used to announce major version releases: mummer-users@lists.sourceforge.net
MUMmer tools

• nucmer $\rightarrow$ (NUCleotide MUMmer) DNA sequence alignment

• promer $\rightarrow$ PROmer (PROtein MUMmer) - all matching and alignment routines performed on the six frame amino acid translation of the DNA input sequence
$ nucmer --prefix=out contigs.fa 23.fasta

$ mummerplot -postscript out.delta
Generate assemblies with velvet with three different hash sizes.

Which assembly produces the best N50 statistic?

Which assembly produces the longest contigs how does that compare to your expected sizes?