Genome Assembly Lab Part2 – Genome Assembly

BCB 5250 Introduction to Bioinformatics II

Spring 2020

Tae-Hyuk (Ted) Ahn
Department of Computer Science
Program of Bioinformatics and Computational Biology
Saint Louis University

SAINT LOUIS
UNIVERSITY
EST. 1818

Lab: Genome Assembly Lab

- Quality control with FastQC
- Data trimming and error correction (optional)
- Genome assembly with Velvet
- Genome assembly with Spades (Homework)
- Validate the assembly with Quast

Assemble reads with Velvet

- Now, we want to assemble our reads to find the sequence of our imaginary *Staphylococcus aureus* bacterium. We will perform a de novo assembly of the reads into long contiguous sequences using the Velvet short read assembler.

- The first step of the assembler is to build a de Bruijn graph. For that, it will break our reads into *k*-mers, *i.e.* fragments of length *k*. Velvet requires the user to input a value of k (*k*-mer size) for the assembly process. Small k-mers will give greater connectivity, but large *k*-mers will give better specificity.

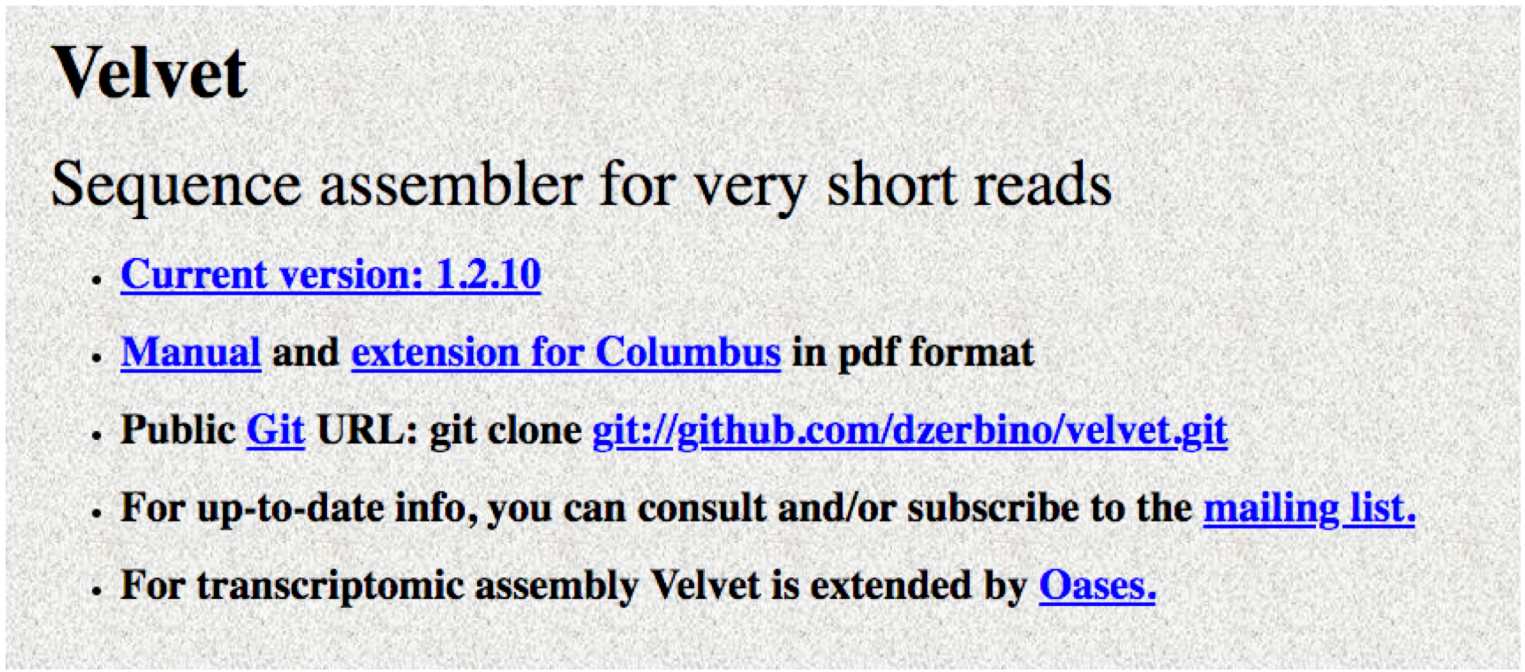
Add a binary path into your PATH

- Normal way for it
  - Open the . **bashrc** file in your home directory (for example, /home/your-user-name/. **bashrc**) in a text editor.
  - **Add** export PATH="your-dir:$PATH" to the last **line** of the file, where your-dir is the directory you want to **add**.
  - Software path is /public/ahnt/courses/bcb5250/genome_assembly_lab/software/velvet_1.2.10/
  - Save the . **bashrc** file.
  - Restart your terminal.

- Command line way for it (if you don't know well how to use Linux text editor)
  - $ echo 'export PATH=/public/ahnt/courses/bcb5250/genome_assembly_lab/software/velvet_1.2.10/:$PATH' >> ~/.bashrc
  - $. ~/.bashrc

Velvet Assembler

- https://www.ebi.ac.uk/~zerbino/velvet/
  - Let us check the manual quickly!



Velvet

Sequence assembler for very short reads

- Current version: 1.2.10
- 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.

Generate Interleaved File

- Currently our paired-end reads are in 2 files (one with the forward reads and one with the reverse reads), but Velvet requires only one file, where each read is next to its mate read. In other words, if the reads are indexed from 0, then reads 0 and 1 are paired, 2 and 3, 4 and 5, etc. Before doing the assembly *per se*, we need to prepare the files by combining them.

- How? You can create a simple Python program for it, right? ☺

- But, use exiting tool: BBmap
- https://jgi.doe.gov/data-and-tools/bbtools/bb-tools-user-guide/reformat-guide/

Generate Interleaved File

- Download and install BBmap or add my BBmap path to your .bashrc
  - $ echo 'export PATH=/public/ahnt/courses/bcb5250/genome_assembly_lab/software/bbmap/:$PATH' >> ~/.bashrc
  - $ ~/.bashrc
- Move to your data directory
- Run the reformat.sh
- $ reformat.sh in1=mutant_R1.fastq in2=mutant_R2.fastq out=mutant_interleaved.fastq

Run velveth

- Go to the labs/genome_assembly/ directory and create a “assembly” directory
- Run velveth tool with the following parameters
  - “Hash Length” to 29
  - Output hash directory name to run_29
  - Input data to ../mutant_interleaved.fastq
  - “file format” to fastq
  - “read type” to shortPaired reads

  - Finally:
  - $ velveth run_29 29 -fastq -shortPaired -interleaved ../data/mutant_interleaved.fastq

Run velvetg

- $ velvetg run_29/
- Two files are generated:
  - A “Contigs” file:

    This file contains the sequences of thecontigs. In the header of each contig, a bit of information is added:
  - the k-mer length (called “length”): For the value of k chosen in the assembly, a measure of how many k-mers overlap (by 1 bp each overlap) to give this length
  - the k-mer coverage (called “coverage”): For the value of k chosen in the assembly, a measure of how many k-mers overlap each base position (in the assembly).

Validate assembly with Quast

- QUAST evaluates genome assemblies
- http://quast.sourceforge.net/index.html
- Quast is already installed in our system.
- Run QUAST with the result

```
$ cd assembly/run_29
$ quast -h
$ quast contigs.fa
```

- How many contigs? Total length? (base? >500bp?) What is N50? Maximum?

Coverage Distribution

- Briefly, the Kmer coverage (and much more information) for each contig is stored in the file stats.txt and can be used with R to visualize the coverage distribution.
- Take a look at the stats.txt file, start R, load and visualize the data using the following commands:

```
$ R
> library(plotrix)
> data <- read.table("stats.txt", header=TRUE)
> jpeg('stats_dist.jpg')
> weighted.hist(data$short1_cov, data$lgth, breaks=0:50)
> dev.off()
```

- If plotrix library is not installed, install it in your local.

Improve the results

- The weighted histogram suggests to me that the expected coverage is around 14 and that everything below 6 is likely to be noise.
- Some coverage is also represented at around 20, 30 and greater 50, which might be contamination or repeats (depending on the dataset), but at the moment this should not worry you.
- To see the improvements, rerun velvetg first with -cov_cutoff 6 and after checking the N50 use only / add -exp_cov 14 to the command line option.
- For the moment focus on the two options -cov_cutoff and -exp_cov.
- Clearly -cov_cutoff will allow you to exclude contigs for which the kmer coverage is low, implying unacceptably poor quality.
- The -exp_cov switch is used to give velvetg an idea of the coverage to expect.

Improve the results

```
$ cd run_29
$ cp contigs.fa contigs.fa.org
$ cd ..
$ velvetg run_29 -cov_cutoff 6 -exp_cov 14
$ cd run_29
$ quast contigs.fa
```

- What is the N50 with -cov_cutoff 6 -exp_cov 14? Compare it with previous results.
  - 5811 vs 132140 ?? Hugely different!

K-mers and optimization

- Now that we would like to see the effect of k-mer size on the assembly, we will run the Velvet to choose the best k-mer size for us.
- We will use a few of K-mers and determine the best k-mer value to use using the “n50”.
- Can you try k-mer size 51 as
  - $ velveth run_51 51 -fastq -shortPaired -interleaved ../data/mutant_interleaved.fastq

```
[0.000000] Velvet can't handle k-mers as long as 51! We'll stick to 31 if you don't mind.
[0.018876] Reading FastQ file ../data/mutant_interleaved.fastq;
[0.242154] 24960 sequences found
[0.242174] Done
[0.257574] Reading read set file run_51/Sequences;
[0.263772] 24960 sequences found
[0.290577] Done
[0.290596] 24960 sequences in total.
[0.292350] Writing into roadmap file run_51/Roadmaps...
[0.338986] Inputting sequences...
[0.339000] Inputting sequence 0 / 24960
[0.943197]  === Sequences loaded in 0.604216 s
[0.950050] Done inputting sequences
[0.950063] Destroying splay table
[0.970698] Splay table destroyed
velveth: Word length 33 greater than max allowed value (31).
Recompile Velvet to deal with this word length.: No such file or directory
```

Lab Homework!

- Complete the lab.
- Run “Spades” for the genome assembly of lab2.
  - Run Quast of the assembly.
  - Compare the results with Velvet.
- Report the QUAST results with your comments.
- Why does velvet require an odd-length kmer? Search velvet manual and describe it.