Genome Assembly Lab

BCB 5200 Introduction Bioinformatics I

Fall 2017

Tae-Hyuk (Ted) Ahn

Department of Computer Science
Program of Bioinformatics and Computational Biology
Saint Louis University

SAINT LOUIS UNIVERSITY™

EST. 1818

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

Prepare the lab

- Make your sub-directory for this lab as me:

```
[ahnt@hopper:~/Course/BCB5200/genome_assembly_lab] $
```

- Download the data
  - The data you will examine is from *Staphylococcus aureus* USA300 which has a genome of around 3MB . The reads are Illumina and are unpaired, also known as single-end library.

```
$ mkdir data

$ cd data

$ wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR022/
SRR022825/SRR022825.fastq.gz

$ wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR022/
SRR022823/SRR022823.fastq.gz
```

Prepare the lab

- Make your sub-directory for this lab as me:

```
[ahnt@hopper:~/Course/BCB5200/genome_assembly_lab] $
```

- Download the data
  - If ftp is blocked, then try http instead of ftp

```
$ wget http://ftp.sra.ebi.ac.uk/vol1/fastq/SRR022/
SRR022825/SRR022825.fastq.gz
```

```
$ wget http://ftp.sra.ebi.ac.uk/vol1/fastq/SRR022/
SRR022823/SRR022823.fastq.gz
```

Quality Control of the Data

- FastQC: Quality control check

  https://www.bioinformatics.babraham.ac.uk/projects/fastqc/

  - FastQC aims to provide a simple way to do some quality control checks on raw sequence data coming from high throughput sequencing pipelines. It provides a modular set of analyses which you can use to give a quick impression of whether your data has any problems of which you should be aware before doing any further analysis.

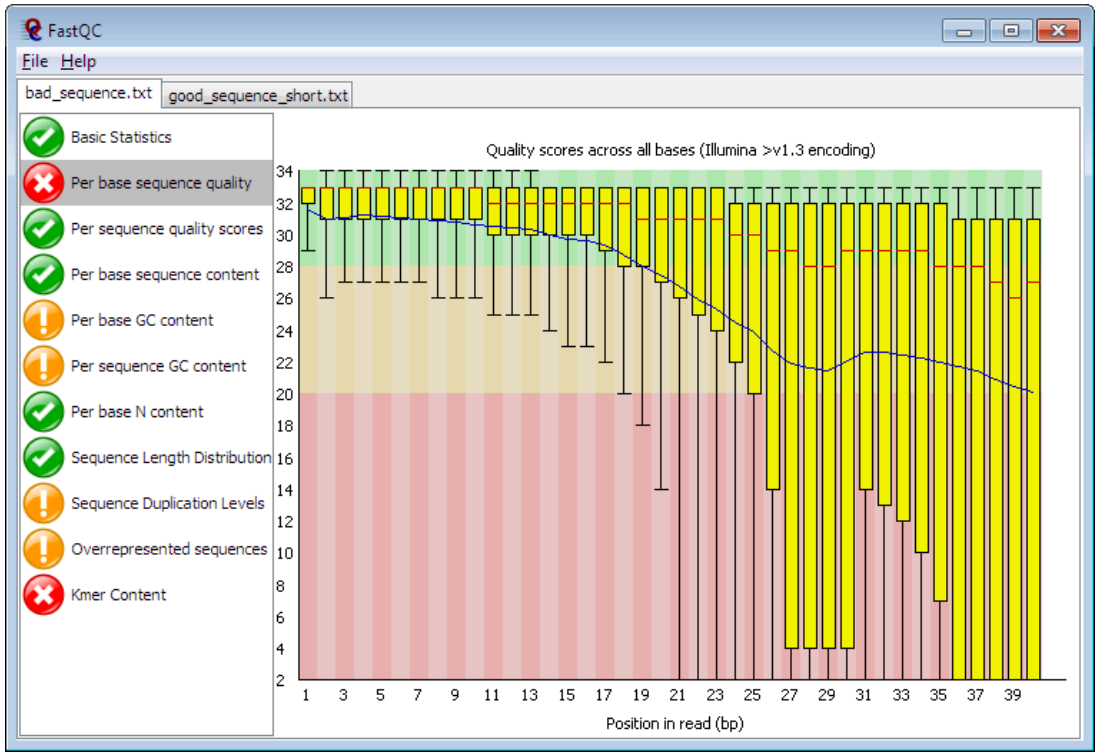
FastQC

Function	A quality control tool for high throughput sequence data.
Language	Java
Requirements	A suitable Java Runtime Environment The Picard BAM/SAM Libraries (included in download)
Code Maturity	Stable. Mature code, but feedback is appreciated.
Code Released	Yes, under GPL v3 or later.
Initial Contact	Simon Andrews
Download Now	

Quality Control of the Data

The main functions of FastQC are

- Import of data from BAM, SAM or FastQ files (any variant)
- Providing a quick overview to tell you in which areas there may be problems
- Summary graphs and tables to quickly assess your data
- Export of results to an HTML based permanent report
- Offline operation to allow automated generation of reports without running the interactive application



Quality Control of the Data

- Run FastQC

```
$ fastqc -h
$ fastqc SRR022823.fastq.gz SRR022825.fastq.gz
```

- Check the html results

Trim, Filter, and Error Collection

Trim and filtering:

- FASTX Tool Kit: http://hannonlab.cshl.edu/fastx_toolkit/
- Sickle: https://github.com/najoshi/sickle
- BBTools – BBDuk: https://jgi.doe.gov/data-and-tools/bbtools/bb-tools-user-guide/bbduk-guide/

Error correction:

- BBTools – Tadpole: https://jgi.doe.gov/data-and-tools/bbtools/bb-tools-user-guide/tadpole-guide/
- For long sequencing, check the recently published papers and archive such as http://www.pacb.com/asset_tags/error-correction/

Assembler - Velvet

- Velvet is probably the most popular assembler.
- It works well with any sort of genomic Illumina data.
- However, Velvet uses quite a lot of memory (will need several hundreds of GB of RAM for > 100 Mbp genomes).
- The bacterial dataset we are using here is small, so Velvet will use several GB of RAM.
- Website: http://www.ebi.ac.uk/~zerbino/velvet/
- Manual: http://www.ebi.ac.uk/~zerbino/velvet/Manual.pdf
- Download into software directory (create directory) in the assembly lab

Install Velvet

- Decompress tgz (or tar.gz)

```
$ tar zxvf velvet_1.2.10.tgz
$ cd velvet_1.2.10/
$ make
```

- By default, hash-lengths are limited to 31bp, but you can push up this limit by adjusting the MAXKMERLENGTH parameter at compilation time:

```
$ make clean
$ make 'MAXKMERLENGTH=57'
```

Test velvet

```
$ ./velveth

$ ./velvetg

$ ./velveth sillyDirectory 21 -shortPaired data/test_reads.fa

$ ./velvetg sillyDirectory

$ less sillyDirectory/stats.txt

$ less sillyDirectory/contigs.fa
```

Run velvet for the data

- Now run velveth for the reads in SRR022825.fastq.gz and SRR022823.fastq.gz using the following options:
  - De Bruijn graph k-mer of 25
  - output directory called run_25

```
$ cd ~/Course/BCB5200/genome_assembly_lab

$ mkdir assembly

$ cd assembly

$ ../software/velvet_1.2.10/velveth run_25 25 -
fastq.gz -short ../data/SRR022823.fastq.gz ../
data/SRR022825.fastq.gz

$ ../software/velvet_1.2.10/velvetg run_25
```

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_25
$ quast -h
$ quast contigs.fa
```

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

```
$ cp run_25/contigs.fa run_25/contigs.fa.old
$ ../software/velvet_1.2.10/velvetg run_25 -cov_cutoff 6 -exp_cov 14
$ cd run_25
$ quast contigs.fa
```

- What is the N50 with -cov_cutoff 6 -exp_cov 14? Compare it with previous results.