Genome assembly tutorial

Sebastian Schmeier

Institute of Natural and Mathematical Sciences
Massey University Auckland, New Zealand
http://sebscientific.org
s.schmeier@gmail.com

2015-07-20

Contents

Genome assembly tutorial

0. Preface

In this genome assembly tutorial we will use our skill on the command-line interface to deal with the taks of a genome assembly. You will learn how to judge the sequencing quality of a next generation sequencing (NGS) run and assemble the genome based on a collection of short sequences.

You will encounter some **To-do** sections at times. Write the solutions and answers into a text-file.

A PDF-version of this tutorial can be downloaded here or here.

1. Learning outcomes

1. Being able to compute, investigate and evaluate the quality of sequence data from a sequencing experiment.
2. Being able to compute, interpret and evaluate a whole genome assembly.
3. Being able to judge the quality of a genome assembly.

2. The data

Get the data

You can download the data-file here (alternative link). I will also bring the data on a USB drive, please copy it onto your system should the download not work.

The data is a down-sampled (randomly selected) small portion of the original sequencing data-set. This has been done because the amount of data produced was too high for the exercises today. Also, the original data was paired-end data, thus we had two files, one for each end. The paired-data here was already combined into one file.

Investigate the data

Make use of your newly developed skills on the command-line to investigate the files in two folders.

To-do

1. Unzip the data using `gzip`.
2. Use the command-line to get some ideas about the file.
3. What kind of files are we dealing with?
4. How many sequence reads are in the file?

3.0 Quality assessment of sequencing reads (SolexaQA++)

To assess the sequence read quality of the Illumina run we make use of a program called SolexaQA. This was originally developed to work with Solexa data (since bought by Illumina), but long since working with Illumina data. It produces nice graphics that intuitively show the quality of the sequences. it is alos able to dynamically trim the bad quality ends off the reads.

From the webpage:

SolexaQA calculates sequence quality statistics and creates visual representations of data quality for second-generation sequencing data. Originally developed for the Illumina system (historically known as "Solexa"), SolexaQA now also supports Ion Torrent and 454 data.

Download/install SolexaQA++ (hopefully not necessary)

Download SolexaQA from the developer webpage here.

Note! Emergency link **here** (alternative link).

```
# unzip the archive
$ unzip SolexaQA++_v3.1.3.zip
# make the file executable
$ chmod a+x SolexaQA++_v3.1.3/Linux_x64/SolexaQA++
# copy the executable to my working directory
$ cp ./SolexaQA++_v3.1.3/Linux_x64/SolexaQA++ .
# run the program
$ ./SolexaQA++
# should now run the program
```

Understand SolexaQA++

SolexaQA++ has three modes that can be run. Type:

```
$ ./SolexaQA++
```

```
SolexaQA++ v3.1.3
Released under GNU General Public License version 3
C++ version developed by Mauro Truglio (M.Truglio@massey.ac.nz)

Usage: SolexaQA++ <command> [options]

Command: analysis   quality analysis and graphs generation
         dynamictrim   trim reads using a chosen threshold
         lengthsort sort reads by a chosen length
```

The three modes are: `analysis`, `dynamictrim`, and `lengthsort`

`analysis` - the primary quality analysis and visualization tool. Designed to run on unmodified FASTQ files obtained directly from Illumina, Ion Torrent or 454 sequencers.

`dynamictrim` - a read trimmer that individually crops each read to its longest contiguous segment for which quality scores are greater than a user-supplied quality cutoff.

`lengthsort` - a program to separate high quality reads from low quality reads. LengthSort assigns trimmed reads to paired-end, singleton and discard files based on a user-defined length cutoff.

Run SolexaQA++ on untrimmed data

To-do

1. Create a directory for the result-files –> **qa_untrimmed/**.
2. Run SolexaQA++ with the untrimmed data, and submit result-directory **qa_untrimmed/**.
3. Investigate the result-files in **qa_untrimmed/**.
4. Compare your results to these examples of a particularly bad run (*Figure 1-4*, taken from http://solexaqa.sourceforge.net/).
5. What elements in these figures (Figure1-4) show that we are dealing with a bad run?

Hint! Should you not get 1 and/or 2 it right, try these commands here (alternative link).

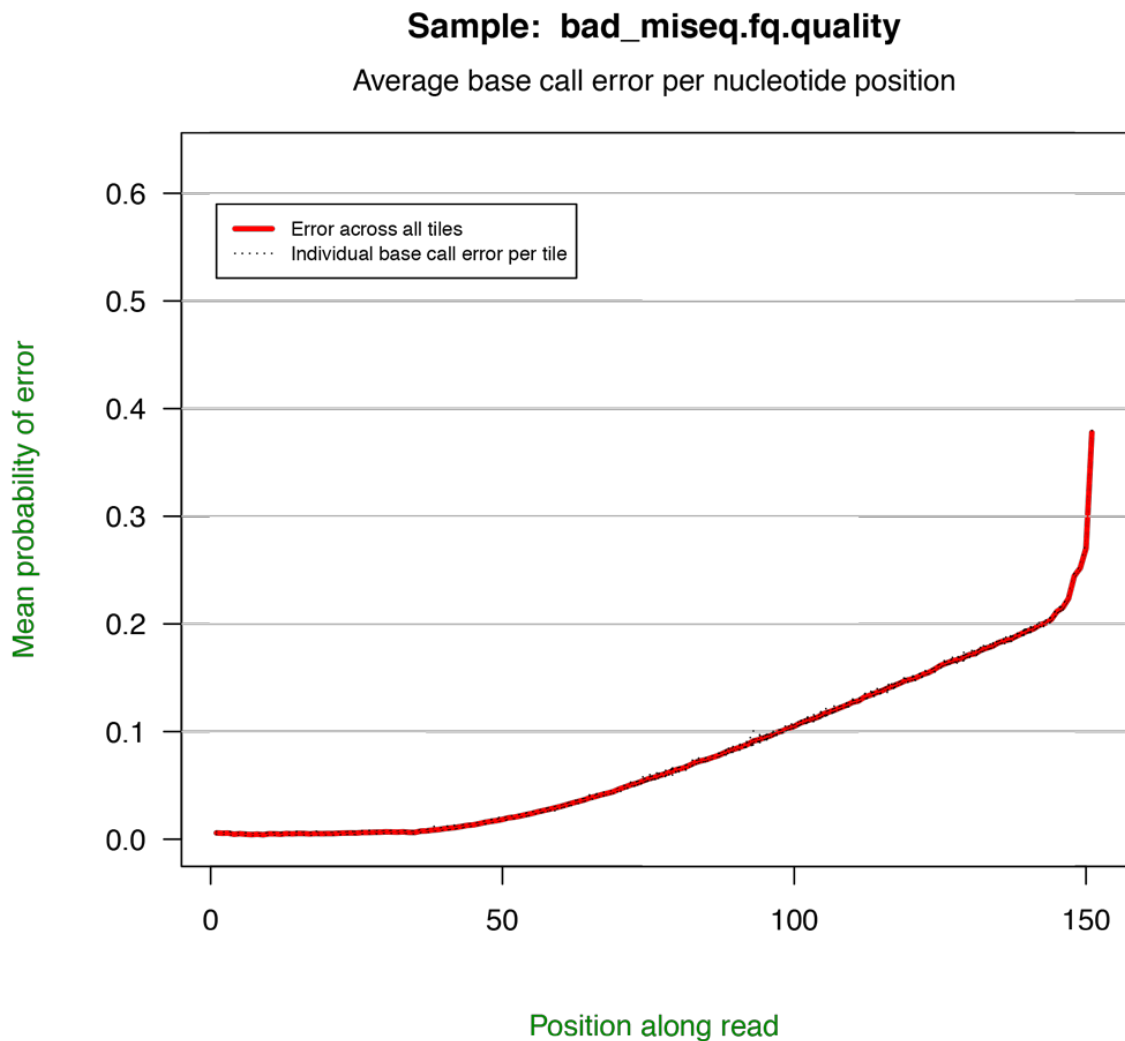# Sample: bad_miseq.fq.quality

Average base call error per nucleotide position

Figure 1: SolexaQA++ quality plot along reads

Sample: bad_miseq.fq.segments

p cutoff = 0.05

Sum of the segments = 1
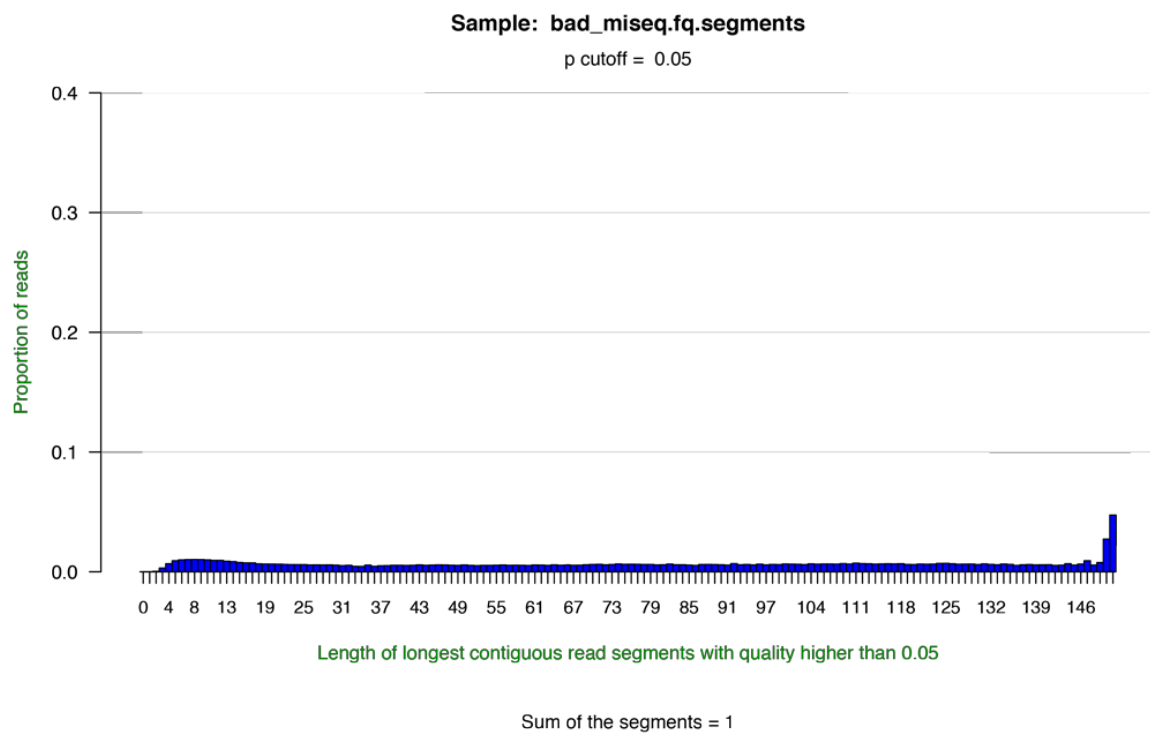
Figure 2: SolexaQA++ histogram plot.

Sample: bad_miseq.fq.segments

p cutoff = 0.05
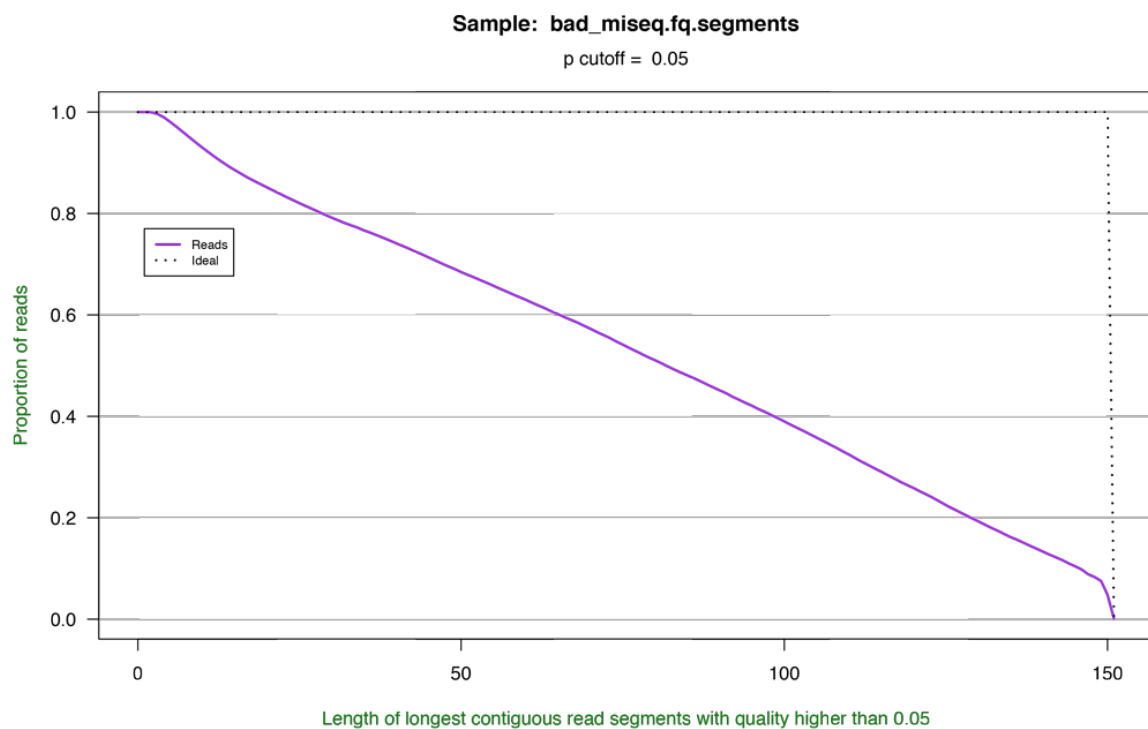

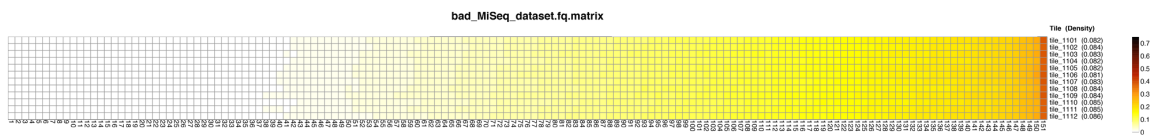Figure 3: SolexaQA++ cumulative plot.

Figure 4: SolexaQA++ quality heatmap

Dynamic trim the data

Despite what you may have found out about the untrimmed data, it is a good idea to trim the data before further analyses.

To-do

1. Create a directory for the result-files –> **qa_toTrimmed/**
2. Use SolexaQA++ to trim the reads based on quality and a probability cutoff of 0.01.

Hint! Should you not get 1 and/or 2 it right, try these commands here (alternative link).

Run SolexaQA++ on trimmed data

To-do

1. Create a directory for the result-files –> **qa_trimmed/**.
2. Do the quality assessment again with the trimmed data-set.
3. Compare the results in **qa_trimmed/** to the untrimmed results in **qa_untrimmed/**.

Hint! Should you not get 1 and/or 2 it right, try these commands here (alternative link).

3.1 Quality assessment of sequencing reads (FastQC)

Download/install FastQC (might already be installed)

Download FastQC from the developer webpage here.

Note! Emergency link here.

```
# Download file on the command-line
$ wget http://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc_v0.11.3.zip
# Unzip the archive
$ unzip fastqc_v0.11.3.zip
# copy the program to the working directory
$ cp ./FastQC/fastqc .
# run the program
$ ./fastqc --help
# should now run the program
```

```
                    FastQC - A high throughput sequence QC analysis tool

SYNOPSIS

        fastqc seqfile1 seqfile2 .. seqfileN

    fastqc [-o output dir] [--(no)extract] [-f fastq|bam|sam]
           [-c contaminant file] seqfile1 .. seqfileN

DESCRIPTION

    FastQC reads a set of sequence files and produces from each one a quality
    control report consisting of a number of different modules, each one of
    which will help to identify a different potential type of problem in your
    data.

    If no files to process are specified on the command line then the program
    will start as an interactive graphical application.  If files are provided
    on the command line then the program will run with no user interaction
    required.  In this mode it is suitable for inclusion into a standardised
    analysis pipeline.
```

Understand FastQC

FastQC is a very simple program to run that provides similar and additional information to SolexaQA++.

From the webpage:

> 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.

The basic command looks like:

```
$ fastqc -o result-dir/ input-file.[txt/fa/fq] ...
```

- `-o result-dir/` is the directory where the result files will be written
- `input-file.[txt/fa/fq]` is the sequence file to analyze, can be more than one file.

The result will be a HTML page per input file that can be opened in a web-browser.

Run FastQC on the untrimmed and trimmed data

To-do

1. Create a directory for the results –> **fastqc_results/**

2. Run FastQC with both files
3. Compare the two result-files in a browser
4. Compare your results to these examples (*Figure 5-6*) of a particularly bad run (taken from `http://www.bioinformatics.babraham.ac.uk/projects/fastqc/`.
5. What elements in these figures (Figure1-4) show that we are dealing with a bad run?

Hint! Should you not get it right, try these commands here (alternative link).

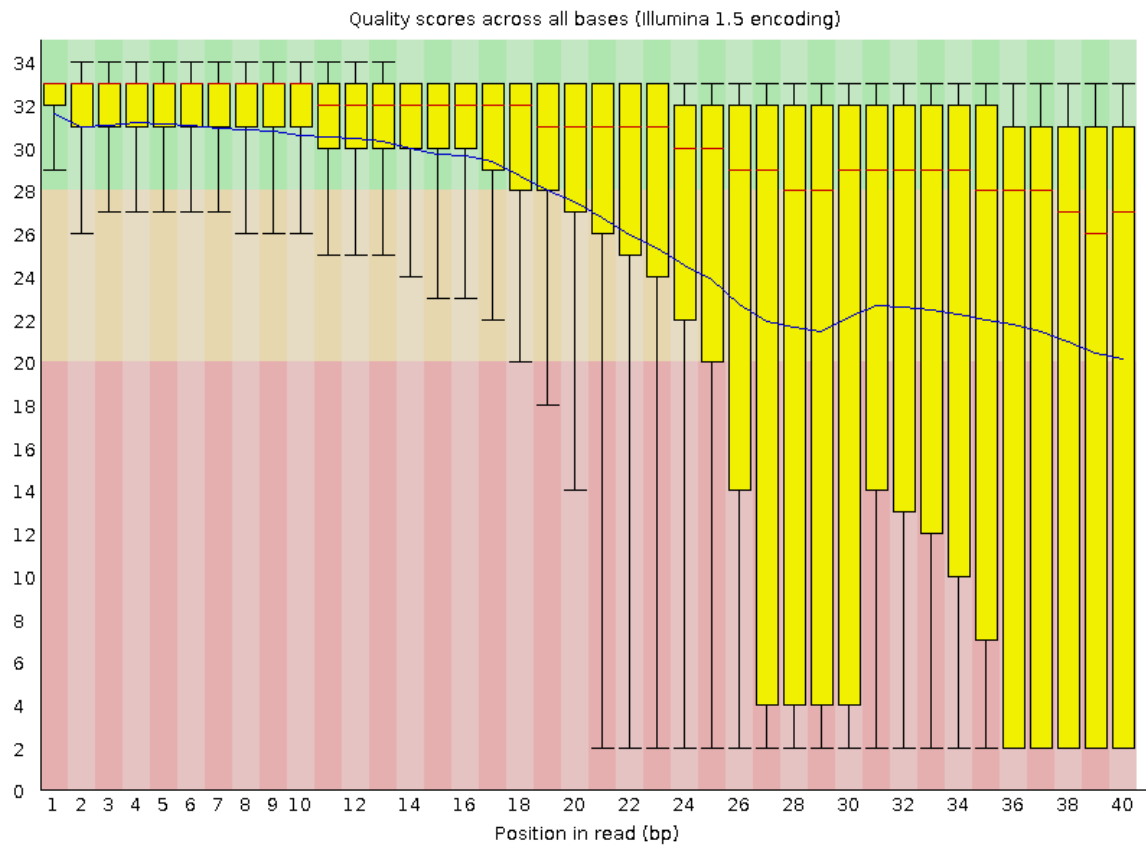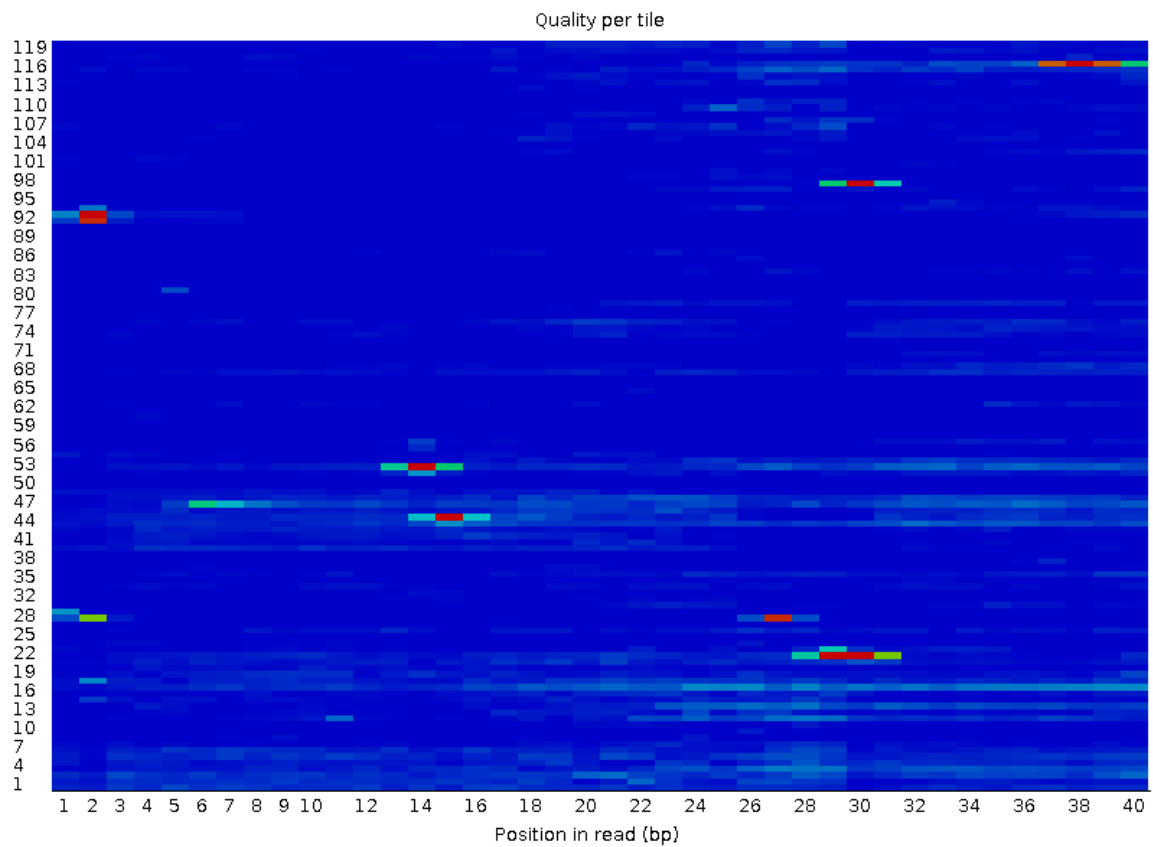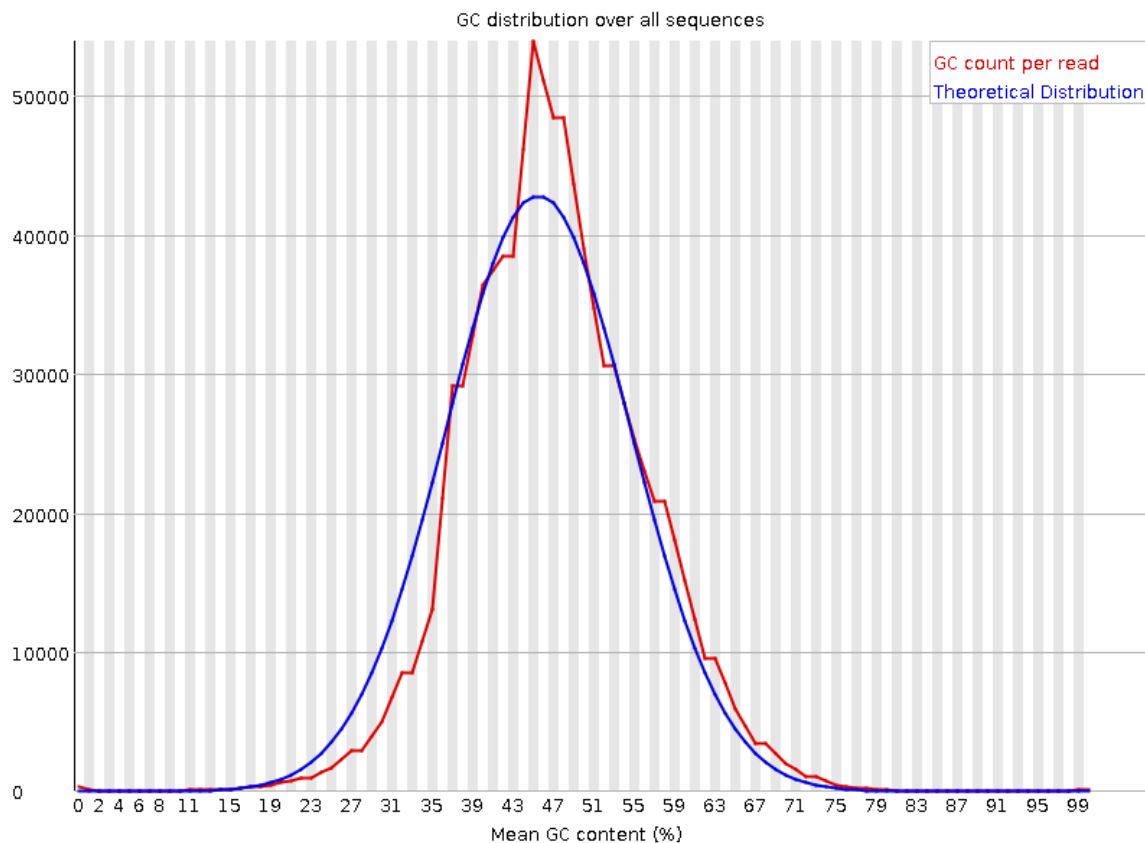*Figure 5: Quality score across bases.*

Figure 6: Quality per tile.

Figure 7: GC distribution over all sequences.

4. Whole genome assembly (velvet)

Velvet is a de Bruijn graph assembly program. It is developed with the aim of assembling very short reads like in our case.

From the webpage:

> Velvet is a de novo genomic assembler specially designed for short read sequencing technologies, such as Solexa or 454. Velvet currently takes in short read sequences, removes errors then produces high quality unique contigs. It then uses paired-end read and long read information, when available, to retrieve the repeated areas between contigs.

Download/install Velvet (might already be installed)

Download Velvet from the developer website here.

Note! Emergency download-link **here** (alternative link).

```
# unzip
$ gunzip velvet_1.2.10.tgz
# extract the archive
```

```
$ tar xvf velvet_1.2.10.tar
# change directory
$ cd velvet_1.2.10/
# compile program
$ make 'MAXKMERLENGTH=63'
# this will take a few seconds
# we have now two programs: velveth and velvetg


# copy these to your home directory
```

Understanding Velvet

Velvet uses two different programs sequentially to achieve the assembly. You have to use them one at a time. You should familiarize yourself with the structure of the program calls. Once you understand this it is very easy to change things around.

`velveth`

This program is a hashing program. It basically prepares your data for the assembler `velvetg`. An example program call looks like:

```
# Running velveth without parameters gives you a help page
$ ./velveth
# Here a complete example call
$ ./velveth results_directory/ 63 -fastq -shortPaired input_seqs.fastq
```

- `./velveth` the program inlcuding the full path where the program is located
- `result_directory/` the directory where you want the results to be saved
- `63` the kmer value
- `-fastq` specifies the input sequence-file type
- `-shortPaired` the type of the reads (here short paired reads)
- `input_seqs.fastq` the file-name of the input data

`velvetg`

This program is the core of velvet and the actual assembler. An example program call looks like this:

```
$ ./velvetg result_directory/ -cov_cutoff 10 -exp_cov 30
      -min_contig_lgth 500 -ins_length 300
```

- `./velvetg` the program including the full path where the program is located
- `result_directory/` the directory where you want the results to be saved
- `-cov_cutoff 10` removal of low coverage nodes AFTER tour bus or allow the system to infer it [do not worry about this]
- `-exp_cov 30` the expected coverage of unique regions or allow the system to infer it [do not worry about this]
- `-min_contig_lgth 500` the minimum size of contig to report back (to exclude tiny ones we only look at contigs >500bp)
- `-ins_length 300` expected distance between two paired-end reads in the respective short-read dataset

Run Velvet with untrimmed data

To-do

1. Make sure you have the untrimmed data and you know where it is.
2. Create a directory for the results –> `velvet_untrimmed/`
3. Run velveth with the data `eli.low10paired.fastq`
4. Run velvetg

Hint! Should you not get it right, try these commands here (alternative link).

Run Velvet with trimmed data

To-do

1. Make sure you have the trimmed data and you know where it is.
2. Create a directory for the results –> `velvet_trimmed/`
3. Run velveth with the data `qa_toTrimmed/eli.low10paired.fastq.trimmed`
4. Run velvetg

Hint! Should you not get it right, try these commands here (alternative link).

Evaluate assemblies

To-do

1. Look at the Log files from trimmed and untrimmed assemblies.
2. Look at the stat.txt files from trimmed and untrimmed assemblies.
3. Look at the contigs.fa files of the trimmed and untrimmed assemblies.
4. What can we say about the the assemblies?
5. How does untrimmed and trimmed compare?
6. What can you say about the trimming procedure in light of assembling sequences?

Visualize assemblies

Bandage

Assembly quality assessment

Quast, download link here.

5.Genome annotation

Genome annotation pipeline RAST

Phage-specific annotation pipeline PHAST

6. What's next?

Next steps could include:

- map all reads to the "new" genome
- look at the aligned reads in a genome viewer
- predict genes in the "new" genome
- overlay gene information with aligned reads in the genome viewer
- etc.

File: index.md - PDF-version: link1 | link2 - Sebastian Schmeier - Last update: 2015-07-20