Q&A

# Reads quality control

* **What is the structure of a FASTQ file?**

A FASTQ file is composed by multiple entries, each of which consists of four lines. The first one is a sequence identifier, with information about the cluster and sequencing run (instrument, run number, flowcell ID, lane, tile...). The second line contains the sequence (nucleotide bases). Afterwards, there’s a + sign that works as a separator. Finally, the last line has Illumina Quality Scores.

Since these are paired-read runs, there are two files per sample: R1 and R2. They can be found in each \*.fastqc.gz file.

* How is the quality of the data stored in the FASTQ files? How are paired reads identified?

Explain Illumina Quality Scores here

Find id for paired reads

* How is the quality of your data?
* What can generate the issues you observe in your data? Can these cause any problems during subsequent analyses?

# Trimming / Reads preprocessing

* How many reads have been discarded after trimming?
* How can this affect your future analyses and results?
* How is the quality of your data after trimming?
* What do the LEADING, TRAILING and SLIDINGWINDOW options do?

# Genome assembly

* **What information can you get from the plots and reports given by the assembler (if you get any)?**

Canu is an assembler that provides a lot of information to the user, both in the form of plots and reports. In its report, it informs the user of how many reads were corrected and trimmed, the amount of contigs and unitigs found and several quality characteristics of them. For example, 4 contigs were produced, no bubbles were found and 3946 sequences were unassembled. Plots for both contigs and unitigs are also generated.

* **What intermediate steps generate informative output about the assembly?**

Canu generates a lot of information about the intermediate steps it performs. Its output produces, for example, a correction and trimming folder with logs about corrected reads. It also generates a folder, called canu-logs, with information about steps performed by this assembler.

* **How many contigs do you expect? How many do you obtain?**

The amount of contigs obtained was expected to be somewhat variable. As Max, one of the teaching assistants of this course, explained in Slack, the assembly is a heuristic process and does not always produce the same output. The authors of the paper obtained two of them, while Canu produced four in my case. This was to be expected.

* **What is the difference between a ‘contig’ and a ‘unitig’?**

A unitig is a verified, or high-confidence contig. Contigs are sets of reads that are put together to cover a continuous, contiguous sequence. Unitigs are an unique assembly of a subset of fragments, there are no other possible choices for how that region could be assembled (ignoring the existence of repeats).

* **What is the difference between a ‘contig’ and a ‘scaffold’?**

A scaffold is a combination of ordered contigs that are connected together, a set that can also include gaps between said contigs. Contigs are, by definition, continuous.

* **What are the k-mers? What k-mer(s) should you use? What are the problems and benefits of choosing a small kmer? And a big k-mer?**

A k-mer is a subset of a sequence, of an assigned length k. According to Canu’s documentation, it is ‘a contiguous sequence of k bases’. Choosing a k-mer depends on the particularities of your data and what information you intend to obtain. Using a big k-mer makes it less likely to generate the exact same k-mer twice, so similar regions like repeats will be better assembled. In the other hand, it also makes it more vulnerable to sequencing errors. Small k-mers are a good idea for analysis of short reads or if there are not many repeats. It is also less computationally expensive.

* **Some assemblers can include a read-correction step before doing the assembly. What is this step doing?**

Canu is one of those assemblers. Its read-correction step consists on a quality assessment of the reads. If any segments are considered to have too poor quality to be assembled, they are trimmed from the reads. This is extremely convenient for the user, since they do not have to perform quality evaluations of the raw data before assembling.

* **How different do different assemblers perform for the same data?**

Only Canu was used for assembly in my analysis. According to the paper, authors used HGAP3 at their sequencing facility, and afterwards they used Gepard to inspect dot plots and Circlator to circularize it. Since I have not performed those steps, I cannot really make an informed comparison.

* **Can you see any other letter apart from AGTC in your assembly? If so, what are those?**

There are no other letters apart from AGTC in my assembly. For example, a possibility would have been N (any nucleotide) or W (indicating A or T, ‘weak’ nucleotides because of their two hydrogen bonds).

# Assembly evaluation

* **What do measures like N50, N90, etc. mean? How can they help you evaluate the quality of your assembly? Which measure is the best to summarize the quality of the assembly (N50, number of ORFs, completeness, total size, longest contig ...)**

NX measurements are quality parameters that give you the length of the shortest contig needed to cover X% of the genome. They are frequently used to evaluate assembly quality (especially N50) since high values would indicate that there is a long average length of contigs. However, what is considered a ‘high’ value depends of the particular case that is under study. Therefore, this parameter should not be used to compare different assemblies (of different genomes) in many cases. Assessing assembly quality can be tricky, and therefore, it is most useful to take into account different parameters and use reference literature of similar organisms.

* **How does your assembly compare with the reference assembly? What can have caused the differences?**

While reading Quast’s report, it was brought to my attention that my assembly contained one misassembly of a very large size (2563357 bp), which was the same as the size of the largest contig. However, other parameters seem to be fine, so it was unlikely that any big mistakes were committed during the assembly. While this can seem confusing at first, it is most likely a consequence of this genome being circular.

* **Why do you think your assembly is better/worse than the public one?**

I do not think that there are extremely significant differences between my assembly and the public one. Most of Quast’s plots show a good overlap between this assembly and the reference. The output from nucmer does not show any big issues either. However, it is true that this assembly could have been improved if I had more expertise.

# Annotation

* **What types of features are detected by the software? Which ones are more reliable a priori?**

Prokka detects genes, CDS (protein coding genes), rRNA (ribosomal RNA), signal peptides, tmRNA (transfer-messenger RNA), tRNA (transfer RNA), repeat regions and other ‘miscellaneous’ RNA. Prokka was designed for prokaryotic genome annotation and uses several highly reliable databases. Therefore, there should be no problem with the reported information. However, rRNA is very conserved, so it could be considered the most reliable feature reported.

* **How many features of each kind are detected in your contigs? Do you detect the same number of features as the authors? How do they differ?**

In my contigs, I detected 2572 CDS, 2635 genes, 6 rRNAs, 112 signal peptides, 1 tmRNA, 48 tRNAs, 1 repeat region and 8 miscellaneous RNA. The authors detected different numbers: 2541 total genes, 2486 CDS and 277 signal peptides. RNA data, however, was the same. The most likely explanation for this is that I have been performing this analysis in 2020, with more updated databases and a more updated version of Prokka than those available when this paper was published.

* **Why is it more difficult to do the functional annotation in eukaryotic genomes?**

Although it is not necessarily so, eukaryotic genomes have a tendency to be larger, and therefore, there is more room for error (statistically speaking). It is also common for eukaryotic genomes to contain many repeats, which complicates maintaining assembly quality, and therefore, annotation too. However, the most relevant difference in comparison to prokaryotic genomes is gene splicing, post-transcriptional modifications in which a single gene can code for multiple proteins.

* **How many genes are annotated as ‘hypothetical protein’? Why is that so? How would you tackle that problem?**

Prokka reported X hypothetic proteins. The reason why is because it is unable to find a function for them, since there are no similar genes reported. To solve this problem,

* **How can you evaluate the quality of the obtained functional annotation?**

A way of doing this, although not very practical, would be to manually revise the functional annotation and compare it to published information from databases/reported annotated genomes from similar organisms. For example, gene function could also be studied in wet lab experiments.

* **How comparable are the results obtained from two different structural annotation softwares?**

I did only use Prokka for annotation, since I did not manage to run eggNOGmapper successfully. However, if we consider a scenario in which two different structural annotation softwares were used, it would be important to evaluate if the numbers reported for different features were similar. If possible, annotations could be manually studied for consistency.

# Homology search

* **How relevant is the output format that you choose?**

There are several output formats available, depending on your particular needs or the software you will use afterwards. For example, I opted for the tabular format, since I needed it in order to use ACT (other formats were not recognized by it).

* **How do the resulting hits vary when you change the minimum e-value?**

The minimum e-value allows users to ‘filter out’ any non-relevant results, since BLAST could include in its output poorly aligned hits. The lower the e-value, the more significant the result is. In my case, I did not set the minimum e-value manually, since I blasted against a similar genome *(L. ferrooxidans).* However, it is important to keep in mind that when blasting against an entire database, e-value is extremely important to obtain quality results.

* **How is the alignment score calculated?**

The alignment score is calculated considering substitution and gap scores. BLAST uses BLOSUM matrices by default, although PAM is also an available option. Said matrices are used to compare one amino acid in one sequence to another, and determine how ‘likely’ it is that the substitution took place (it indicates divergence or similarity). Gap scores are determined by opening gaps in the alignment, and the penalty is extended by extending the gaps. The larger the gaps in the alignment, the worse the score will be. The combination of these two factors determines the alignment score.

* **How important is the number of threads when you blast against a database, or against a particular sequence?**

As in any computational process, a higher number of threads allow BLAST to use more resources and to work quicker. When blasting against a small local database or with small sequences, fewer threads are necessary.

# Mapping

* What percentage of your reads map back to your contigs? Why do you think that is?
* What potential issues can cause mRNA reads not to map properly to genes in the chromosome? Do you expect this to differ between prokaryotic and eukaryotic projects?
* What percentage of reads map to genes?
* How many reads do not map to genes? What does that mean? How does that relate to the type of sequencing data you are mapping?
* What do you interpret from your read coverage differences across the genome?
* Do you see big differences between replicates?

# Post-mapping analyses

* What is the structure of a SAM file, and how does it relate to a BAM file?
* What is the structure of vcf and bcf files?
* How many SNPs and INDELs do you get?
* How is the quality of those variants?
* What is the difference between the variant quality, the mapping quality and the fastq quality?
* How are these variants distributed along the genome?

# Read counting

* What is the distribution of the counts per gene? Are most genes expressed? How many counts would indicate that a gene is expressed?

# Expression analyses

* If your expression results differ from those in the published article, why could it be?
* How do the different samples and replicates cluster together?
* What effect and implications has the p-value selection in the expression results?
* What is the q-value and how does it differ from the p-value? Which one should you use to determine if the result is statistically significant?
* Do you need a normalization step? What would you normalize against? Does DESeq do it?
* What would you do to increase the statistical power of your expression analysis?