Metagenomic Assembly Lab

Duration: ~ 1 hour

Sergey Nurk, NIH

Meet the data
Even though synthetic communities are much simpler than typical real ones, they can be very useful for testing (especially if designed to contain challenging cases).
Today we will be using the metagenomic mock community from JGI.
"Our synthetic microbial community BMock12 consists of 12 bacterial strains with genome sizes spanning 3.2–7.2 Mbp, 40–73% GC content, and 1.5–7.3% repeats."

The two datasets for BMock12 we will be considering today are:

- Illumina 2x150bp (avg IS 302.70) avg read identity 99.8 (SRA accession SRR8073716)
- ONT variable read length, with average identity 85.9 (SRA accession SRR8073713)

All the genomes used to build the community have finished reference genomes (for more info see Supplementary Tables from the original publication).

Illumina and ONT datasets were downsampled to 10% and 20% of original depths respectively (hence the naming illumina/SRR8073716_(1/2).0p1.fastq.gz and ont/SRR8073713.0p2.fastq.gz).
5% downsampled Illumina dataset (illumina/SRR8073716_(1/2).0p05.fastq.gz) is also present in the folder, but will be ignored this time.
NB. The resulting Illumina dataset has 10 times more total sequenced nucleotides than ONT.

Before assembly

Before assembling Illumina data it is important to check reads with FastQC.
Crucial thing to look for -- adapter read-throughs.
Luckily I've checked the data before and we are fine :)

Optional task: run FastQC on Illumina dataset and check the reports

Assembly

Since metagenomic assembly of even a mock dataset might take some time (especially if all you have is 8Gb of RAM and 4 cores), all the data has **already been preassembled** with

MEGAHIT v1.2.9, **metaSPAdes** v3.13.1 and **metaFlye** v2.6 (see illumina/{megahit,metaspades}_0p1 and ont/flye_0p2 folders).
Note that only some files have been left in the folders to save space!

Command lines that I used were:

DO NOT
*megahit -t 12 -m 80000000000 -1 <left> -2 <right> -o <out> (*Yes, megahit memory limit is set in bytes :))
spades.py --meta -t 12 -m 80 -1 <left> -2 <right> -o <out> (-m 80 gives spades 80G of RAM)
flye --meta --min-overlap 1000 --nano-raw <reads> --genome-size 60M --threads 12 --out-dir <out>

Task Figure out the maximal kmer-length value used by MEGAHIT and metaSPAdes respectively.

QUAST

Let's start QUAST assessment, using available reference genomes.

mkdir analysis; cd analysis
ln -s ../ont/flye_0p2/assembly.fasta flye.fasta
ln -s ../illumina/metaspades_0p1/scaffolds.fasta metaspades.fasta
ln -s ../illumina/megahit_0p1/final.contigs.fa megahit.fasta
*metaquast.py -t 4 -o metaquast -R ../refs/ *.fasta*

NB. While metaQUAST has a feature of finding relevant reference genomes (or at least had it before...), please DO NOT use it! If you will ever need to detect reference genomes -- check out this approach.

Optional discussion. metaQUAST's strategy was based on pulling the genomes from NCBI database based on the predicted ribosomal RNA genes in assembled contigs. Why do you think this might not be the best idea?

While QUAST is running (might take about 30 minutes), let's go visualize some assembly graphs.

Analysis of assembly graphs

We will focus on MEGAHIT and Flye assemblies.

Let's start with MEGAHIT. As you already probably know MEGAHIT does not produce the assembly graph as a default output. But it can be obtained by the following command.

cd illumina/megahit_0p1/
megahit_toolkit contig2fastg 141 intermediate_contigs/k141.contigs.fa > megahit.fastg

Now let's open megahit.fastg in Bandage and draw the graph.

Here we will use the graph to illustrate typical metagenomic assembly issues and properties, but a cool example of how assembly graphs can be used to help in improving MAGs and getting biological insights see this blogpost, accompanying this study.

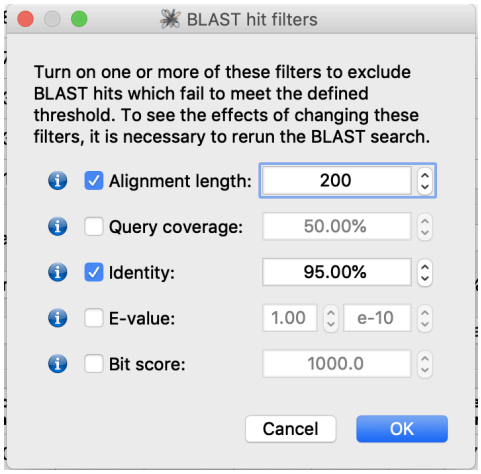
One of the MEGAHIT's advantages over SPAdes is that its contigs are exactly the nodes of its assembly graph.
In SPAdes contig/scaffold corresponds to a (possibly gapped) path in its assembly graph (see "P" records in its output GFA).

Optional task: from the documentation figure out the relationship between scaffolds/contigs in Flye and edges of its assembly graph. Does our Flye assembly have any scaffolds?

NB. It seems that right now there is no easy way to get the mapping between the ids in the MEGAHIT's FastG graph and its *final.contigs.fa*. So if you are planning to use the graph in your work, you will probably have to extract contigs from FastG for consistency for other parts of the analysis. Or wait until this issue is resolved ;)

Task. Look at the swarm of small contigs in the bottom of the picture. What do you think is the reason they are so fragmented? Confirm your suspicion by checking MEGAHIT's depth estimates (and displayed by Bandage) for some of them.

Now let's color the graph according to the reference genomes.
The easy way to do it is to use the integrated visualization of BLAST alignments.
Press "Create/view BLAST search", index the graph, in "Load from fasta file" choose all reference files, set filters as below and press "Run BLAST search".



After it is done check that “Graph display” is set to “BLAST hits (solid)” and let’s see what we got. In the bottom left corner you can switch which alignments are highlighted at the current moment.

Below only the last **three** digits of the genome identifiers are given.

Task: Highlight the subgraph, corresponding to genome 527. Look similar to the graphs you saw in the genome assembly lab, right?

Task: Highlight the subgraph, corresponding to genome 533 (it is represented by two sequences 533A and 533B). What is the major difference with the previous one? Check out coverage depth of the 533 genome.

Task. Look at alignments for genomes 617 and 618. Weird, right? :) And what are all those bubbles? Check out reference files (or reference info from publication) to see if the organism names will help explaining the situation.
Intraspecies microdiversity is a major origin of short and/or incorrect contigs, incomplete MAGs (and headaches...).
Also note how weirdly fragmented the 617/618 subgraph is, despite HIGH coverage. My current hypothesis is that strain differences may be interfering with the local reassembly process in MEGAHIT. But it will need further investigation.

Task. Look at alignments for genomes 697 and 829. Let’s check how similar they are by computing the Average Nucleotide Identity (ANI) index (cool calculator can be found here). Note that even at that similarity level the genomes ended up sharing quite a few 141-mers.
Game: can you spot longer nodes shared by the two genomes? (I have found this one:
)

MEGAHIT's graphs are typically much simpler than ones from metaSPAdes. The reason is the higher value of K that is being used (in this case k=141). By using a cool technique of local reassembly, MEGAHIT is able to increase K almost to the read length without accumulating too many gaps.
Optional task: Open metaspades gfa graph to never open it again :)

Optional problem: Names of SPAdes contigs contain "kmer" coverage -- average multiplicity of the k+1-mers, constituting the sequence in the name of the contig. To get a rough estimate of the per-base coverage (for reasonably long contigs), the value should be multiplied by READ_LENGTH / (READ_LENGTH - K). Why? How do sequencing errors affect the resulting estimate?
Since MEGAHIT's coverage values are also based on the kmer multiplicities, it must be doing this transformation internally.

Now open Flye assembly graph and look how awesome it is.

Task. Which genomes do you think are represented by the only (large) tangled component? Check your suspicion!

Unfortunately de novo assemblies from ONT data have a serious downside: high rate of homopolymer errors in the ONT reads leads to many frame-shift errors within the assemblies. The problem is aggravated by a much lower coverage depth (compared to isolate sequencing projects) for most genomes in metagenomic datasets. For example, initial Flye assembly had ONLY 141 of genes (predicted by prodigal) longer 1500bp (500aa). Polishing with Pilon using Illumina data (min coverage threshold = 7) significantly improved the situation, bringing this number to almost 3.6 thousand. Long read metagenomic analysis is still in its infancy, but super perspective!

Binning and MAG refinement

Not today! :)
If you are interested in the related topics -- check out **http://merenlab.org/posts/**

QUAST II

Now when the metaQUAST is finished explore it's output a bit.

Task. Look at the GF stats for 557 and 567. Note how different they are for Illumina and ONT assemblies. What do you think is the reason? You can take a look at 557 and 567 contigs in the graph.

Note that at the same time the contigs are quite accurate according to QUAST misassemblies report.

metaSPAdes resulted in much fewer misassembly errors than megahit.
Flye did great with exception of 617&618, but note that QUAST's default behavior on the closely related strains might need parameter tweaking as not every strain 'switching' event should probably be considered as a misassembly.

Note the power of long reads wrt NGA50 stats. **Task.** Why do you think some genomes have '-' there and why flye assembly has more of them?

Look at the # mismatch/indel errors.
Elevated number of mismatches in metaSPAdes is a long known problem and a shame to the developers! :(
Importantly, look at the # indels in the Flye assembly! This number goes to **87** (average per 100k) after polishing with Pilon (< 10 indels/100kb for some references).

If you have any time left check out cool Icarus visualizations (get there by the link from report.html or by opening icarus.html).