

Novel DNA Identification -- GVA2020

- [Overview](#)
- [Learning Objectives](#)
- [Relationship to other tutorials](#)
- [Identification of a novel plasmid](#)
 - [Get some data](#)
 - [Map read using bowtie2](#)
 - [Assemble unmapped reads](#)
 - [Compare contigs generated from different assemblies](#)
- [Next steps](#)

Overview

As has been mentioned several times, variants are anything that is different from a reference genome, but large insertions of novel DNA represent something of an unknown unknown. If a read doesn't map is it because its some kind of contamination, or is it because something new has entered into the sample. This tutorial is an attempt to use tools you are already familiar with to identify such novel DNA mutations. As this is a new tutorial any feedback is very welcome and helpful.



Prerequisite required

This tutorial assumes you have already installed spades in the [assembly](#) tutorial. Verify you have done so with the 'spades.py -v' command does not return 3.13.0, need to at least complete enough of that tutorial that spades is installed.

Learning Objectives

1. Extract reads where one or both reads do not map to reference
2. de novo assemble reads

Relationship to other tutorials

1. As presence of adapter can cause problems with assembly, make sure [adapters have been trimmed](#)
2. Reads will be mapped with [bowtie2](#)
3. non-mapped reads will be extracted, consider [checking quality of these reads](#) and possible additional trimming
4. Reads will then be [assembled](#)

None of these other tutorials are required to complete this tutorial, but additional information about individual steps may be found there.

Identification of a novel plasmid

One example of novel DNA being present is when a given sample may have a virus or plasmid associated with a sample. Here we will take a sample known to have a high copy plasmid associated with it, but map the reads against only the genome. Unaligned reads would then be expected to be able to assemble into a plasmid.

Get some data

```
mkdir $SCRATCH/GVA_novel_DNA
cd $SCRATCH/GVA_novel_DNA
cp $BI/gva_course/novel_DNA/* .
ls
```


^ above transfers 2 fastq files and a reference file.

Map read using bowtie2

This is the same process used in the [read mapping tutorial](#), and therefore presented with little comment except to remark on differences. Refer to that tutorial for more in-depth information.

1. Convert reference to fasta

```
module load bioperl
bp_seqconvert.pl --from genbank --to fasta < CP009273.1_Eco_BW25113.gbk > CP009273.1_Eco_BW25113.fasta
```

 Remember to make sure you are on an idev done

For reasons discussed numerous times throughout the course already, please be sure you are on an idev done. It is unlikely that you are currently on an idev node as copying the files while on an idev node seems to be having problems as discussed. Remember the **hostname** command and **showq -u** can be used to check if you are on one of the login nodes or one of the compute nodes. If you need more information or help re-launching a new idev node, [please see this tutorial](#).

Index the reference

```
mkdir bowtie2
module load bowtie/2.3.4
bowtie2-build CP009273.1_Eco_BW25113.fasta bowtie2/CP009273.1_Eco_BW25113
```

3. The following command will take ~5 minutes to complete. Before you run the command execute '**bowtie2 -h**' so while the command is running you can try to figure out what the different options are doing that we did not include in our first tutorial.

Map reads

```
bowtie2 --very-sensitive-local -t -p 48 -x bowtie2/CP009273.1_Eco_BW25113 -1 SRR4341249_1.fastq -2 SRR4341249_2.fastq -S bowtie2/SRR4341249-vsl.sam --un-conc SRR4341249-unmapped-vsl.fastq
```

option	effect
--very-sensitive-local	map in very sensitive local alignment mode
-t	print time info
-p 48	use 48 processors
-x bowtie2/CP009273.1_Eco_BW25113	index
-1 SRR4341249_1.fastq -2 SRR4341249_2.fastq	reads used for mapping
-S bowtie2/SRR4341249-vsl.sam	Sam file detailing mapping
--un-conc SRR4341249-unmapped-vsl.fastq	print reads which do not map to the genome to the file SRR4341249-unmapped-vsl.fastq

The stdoutput of the program listed:

65.74% overall alignment rate

Assemble unmapped reads

This is the same process used in the plasmid assembly portion of the [genome assembly tutorial](#), and therefore presented with little comment except to remark on differences. Refer to that tutorial for more in-depth information.

1. Attempt to assemble with plasmidspades.py which expects to find circularized plasmid type sequences. Command expected to take ~5 minutes

run plasmid spades

```
plasmidspades.py -t 48 -o unmapped_plasmid -1 SRR4341249-unmapped-vs1.1.fastq -2 SRR4341249-unmapped-vs1.2.fastq
```

2. Additionally attempt to assemble with base spades.py command which uses different internal settings to potentially identify different types of novel DNA from our unmapped reads. Command expected to take ~5 minutes.

run plasmid spades

```
spades.py -t 48 -o unmapped_full -1 SRR4341249-unmapped-vs1.1.fastq -2 SRR4341249-unmapped-vs1.2.fastq
```

Compare contigs generated from different assemblies

Recall from [genome assembly tutorial](#) the file we are most interested in is the contigs.fasta file in each output directory (unmapped_plasmid and unmapped_full).

1. 2 for the plasmid and 16 for the full

Where does the answer come from?

```
grep -c "^>" unmapped*/contigs.fasta
```

2. Yes, both contigs detected in the plasmid mode were also detected in the full mode

Where does the answer come from?

```
grep "^>" unmapped*/contigs.fasta
# lists identical lengths and coverages for 2 plasmid contig
```

3. 5441

3170

14 others less than 500bp

Where does the answer come from?

```
grep "^>" unmapped*/contigs.fasta
# same command as above, just focusing on the length value
```

4. The contig that is 3170.

Where does the answer come from?

```
# From above, I stated this was a high copy plasmid, it has a coverage value of 12,698 compared to 98 for the larger contig
```

5. Yes! The actual plasmid locus line stats:

LOCUS GFP_Plasmid_Sequ 3115 bp DNA circular UNA 18-NOV-2013

My thought is that this was raw fastq files that were fed into the assembly, not trimmed files. Leads me to hypothesize that the difference in size is related to the presence of adapter sequence.

Next steps

Here we have presented a proof of concept that unmapped reads can be used to find something that we actually did know was present. We also found something that was even longer that wasn't expected.

Additional questions are:

1. [blast](#). In fact I did just that and identified it as an artifact of sequencing. The contig corresponds to phiX.

Steps to identify phiX

```
copy full 5441 bp of sequence
Go to https://blast.ncbi.nlm.nih.gov/Blast.cgi
large list of results, including vast majority listing phiX or genome assembly/scaffold
```

Why does seeing phiX ([link to screen shot of blast results](#)) tell me that it is an artifact? [phiX is used as a loading control for illumina runs](#) to both tell the difference between a failed run because of bad libraries and a failed run due to poor base diversity.

2. Depends on blast results, how high coverage is compared to genome, gene content
3. viruses, mobile genetic elements, evidence of microbiome
4. [Consider advanced read mapping with multiple references tutorial](#)
5. Similar. The fact that the reference is not as accurate will lower the alignment scores across the board, potentially dropping below thresholds to be able to anchor the match at all. Look deeper at the bowtie2 mapping command where you used `--very-sensitive-local` mode the documentation tells you about tolerated mismatches etc. The more reads you have that don't match, the more novel DNA inserts you are likely to deal with.

[Return to GVA2020](#)