Long-read Data Analysis Workshop

Day 1 tutorials:

PacBio assembly with command line tools, and visualisation tools

Thu 27 - Fri 28 September 2018
UNSW, Sydney

Contents

1

General information

1.1 Workshop trainers

- Scientific coordinators
  - Marc R. Wilkins[1,2,3]
  - Richard J. Edwards[1]
- Workshop Trainers
  - Tim Kahlke[4]
  - Tonia L. Russell[2]
  - Åsa Pérez-Bercoff[1]
  - Xabier Vázquez-Campos[1,3]
  - Nandan Deshpande[1,3]
- Workshop Coordinators
  - Tonia L. Russell[2]
  - Mabel Lum[5]
- Tutorials
  - Tim Kahlke[4]
  - Åsa Pérez-Bercoff[1]
  - Xabier Vázquez-Campos[1,3]
  - Nandan Deshpande[1,3]
  - Melbourne Bioinformatics[6]
- Facilitator
  - Ignatius Pang[1,3]
- IT support
  - Simon Gladman[6]
  - Nic Beatson[7]

1. School of Biotechnology and Biomolecular Sciences, University of New South Wales, Sydney, NSW, Australia
2. Ramaciotti Centre for Genomics, University of New South Wales, Sydney, NSW, Australia
3. NSW Systems Biology Initiative, School of Biotechnology and Biomolecular Sciences, University of New South Wales, Sydney, NSW, Australia
4. Plant Functional Biology and Climate Change Cluster, University of Technology, Sydney, NSW, Australia
5. Bioplatforms Australia
6. Melbourne Bioinformatics, University of Melbourne, Melbourne, VIC, Australia
7. UNSW IT, University of New South Wales, Sydney, NSW, Australia

1.2 Acknowledgements

We acknowledge the assistance of Melbourne Bioinformatics and EMBL-ABR for the hosting of virtual machines and for the sharing of training materials.

We acknowledge Bioplatforms Australia for assistance in the organisation of this workshop.

We acknowledge Prof. Mark Walker as lead investigator on the Bioplatforms Australia Sepsis Project, data of which is used in this project.

The Ramaciotti Centre for Genomics and the Systems Biology Initiative acknowledge the financial support of NCRIS, NSW State Government and UNSW.

1.3 Learning objectives

At the end of this tutorial, we should be able to:

1. Assemble and circularise a bacterial genome from PacBio sequence data.
2. Recover small plasmids missed by long-read sequencing, using Illumina data
3. Explore the effect of polishing assembled sequences with a different data set.
4. Explore the unitig and contig assembly graphs in Bandage. Clean-up the graph to curate the assembly.
5. Compare sequence alignment map (SAM) files made from short- and long-reads respectively when mapped against a reference genome using the Integrative Genomics Viewer (IGV).

1.4 Resources

Tools (and versions) used in this tutorial include:

- Canu v1.7
- EMBOSS v6.6.0.0
  - `infoseq`
  - `sizeseq`
- Circlator v1.5.5
- BWA v0.7.17-r1188
- Samtools v1.7
- SPAdes v3.12.0
- BLAST+ v2.7.1
  - `makeblastdb`
  - `blastn`
- Pilon v1.22
- Bandage v0.8.1
- IGV v2.4.14 (requires Java 8)

1.5 Sample information

The datasets for this and other bacterial strains used in today's practicals have been obtained form the Antibiotic Resistant Sepsis Pathogens project.

The Antibiotic Resistant Sepsis Pathogens Framework Initiative aims to develop a framework dataset of 5 sepsis pathogens (5 strains each) using an integrated application of genomic, transcriptomic, metabolomic and proteomic technologies.

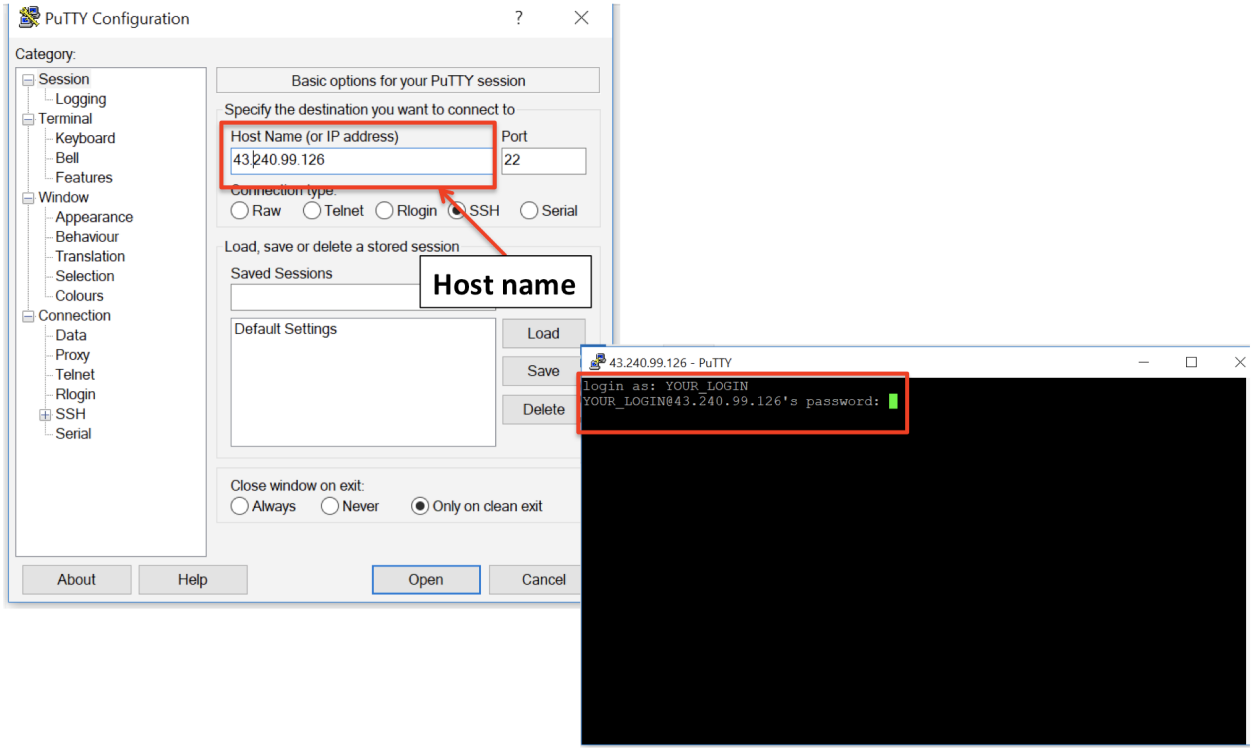

Figure 1.1: PuTTY login. The IP included is only an example.

The main strain used in this tutorial is a gram-negative bacteria called ***Klebsiella pneumoniae*** **strain 25727**. Several strains of this bacteria are known to become opportunistic pathogens which infect humans, and typically causes hospital-acquired infections in immunocompromised patients. Major sites of infection include the lungs, where it causes a type of pneumonia, and urinary tract infections.

1.6 Environment and toolkit set up

This section shows how to set up the environment and load necessary toolkits for the analyses to be run on Day 1.

1. Open the program PuTTY to login to your Virtual machine (VM) instance. This will be your working environment for today.
   - You have been provided with:
     - An individual login: `YOUR_LOGIN`.
     - Host name (or IP address) for your 'Virtual machine' (VM) instance, which is in the format: `123.45.67.789`.
   - Log onto the test server using the `YOUR_LOGIN` user account. See Figure 1.1.
2. Activate the command line tools for Day 1, by excecuting the following command in the terminal window:

```
$ source /mnt/gvl/apps/conda/bin/activate
```

1.7 Data

Once the environment and tools have been set up, let's locate the raw data and the pre-processed dataset:

```
(root) researcher2@long-reads-test-2$ cd workshop_data/
(root) researcher2@long-reads-test-2$ pwd
/mnt/galaxy/home/researcher2/workshop_data
```

> **DO NOT** write on this folder, it contains everything you will need for this tutorial.

You will also need to create your own working directory. Be sure that you work on this directory for the rest of the workshop:

```
$ cd ~
$ mkdir working_dir
$ cd working_dir
$ pwd
/mnt/galaxy/home/researcher2/working_dir
```

> If you are not familiar with this, it will covered in the supplementary session on Intro to Unix and command line

2

Intro to command line and Unix

2.1 Unix File System Tree Structure

This section helps the user understand the Unix File System Tree Structure

2.1.1 Where am I in the file system?

`pwd` - print working directory

```
$ pwd
/mnt/galaxy/home/researcher2
```

2.1.2 View Files

`ls` - list directory contents

To view all the files in your current directory use `ls`. Unix flags modify default behaviour. E.g.

```
$ ls -l
$ ls -a
$ ls -la
```

- `-l` flag gives you long list.
- `-a` stands for "show all".
- `-la` or `-l -a` combines both behaviours.

2.1.3 Changing Directories

`cd` - change directory:

```
$ cd workshop_data/
$ pwd
/mnt/galaxy/home/researcher2/workshop_data
$ cd ../
$ pwd
/mnt/galaxy/home/researcher2/
```

> **Trick:** type `cd w` and hit `Tab`; this is autocomplete.

2.2 Creating and Deleting Things

2.2.1 Objectives

- Create a directory
- Create files using an editor or by copying and renaming existing files.
- Display the contents of a directory using the command line.
- Delete specified files and/or directories.‘

2.2.2 Creating Things

Create a new directory called `thesis` using the command `mkdir` (the command has no output):

```
$ mkdir thesis
```

However, there's nothing in it yet:

```
$ ls -F thesis
```

Change the working directory to thesis using `cd`, then run a text editor called `nano` to create a file called `draft.txt`:

```
$ cd thesis
$ pwd
/mnt/galaxy/home/researcher2/thesis
```

Make a text file using the editor `nano`:

```
$ nano draft.txt
```

> **Note:** Use `Ctrl+X` to exit the editor and return to the shell.

> **Note 2:** `nano` does not save empty files by default.

> Unix documentation often uses the shorthand `^A` to mean `Ctrl+A`.

There's no output from `nano` on the screen after it exits, but ls shows that we have created a file called `draft.txt`.

```
$ ls
draft.txt
```

The commands cat, less and more can all be used to display/view the contents of the file `draft.txt`. `cat` displays the entire files's contents, while more and less display the file's contents one page at a time.

```
$ cat draft.txt
It's not "publish or perish" any more,
It's "share and thrive!"
```

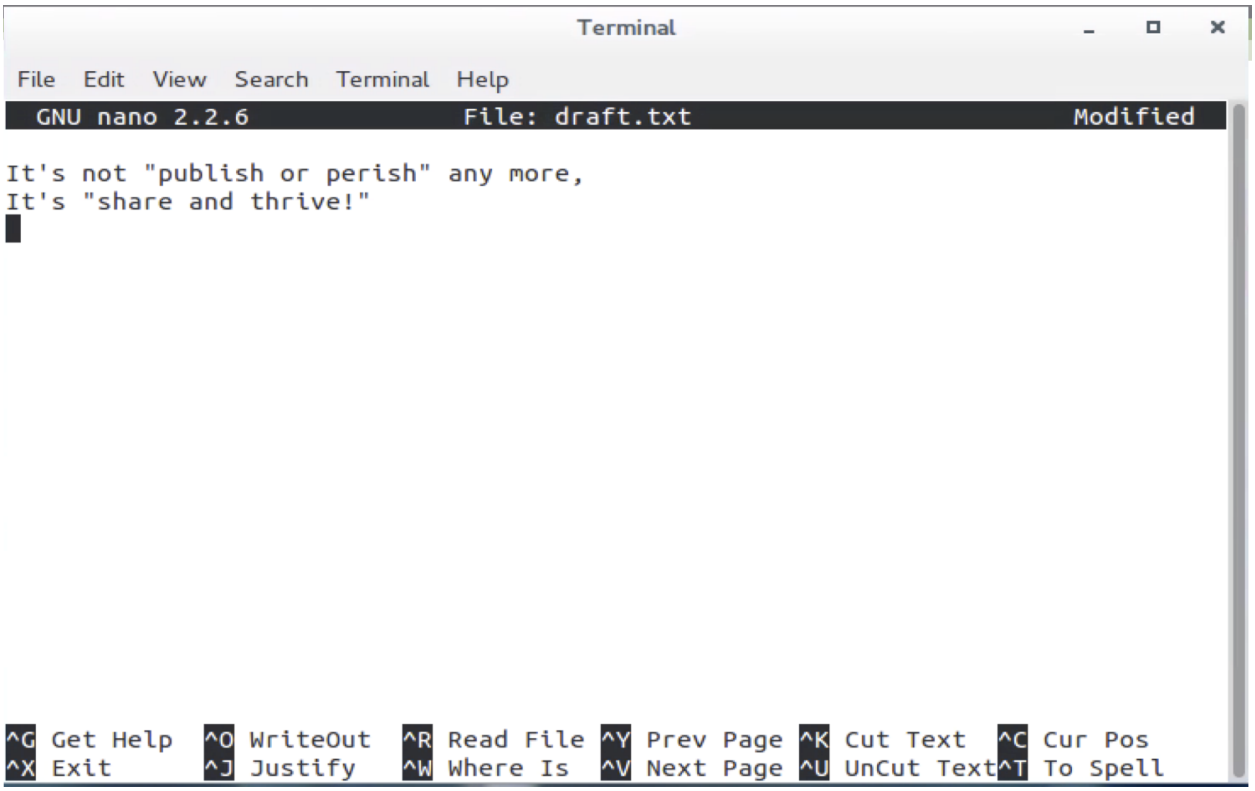

Figure 2.1: `nano` snapshot.

2.2.3 Moving and Copying

Rename `draft.txt` to `quotes.txt` using the command `mv`:

```
$ mv draft.txt quotes.txt
$ ls
quotes.txt
```

The `mv` command can be also be used to move files to other locations in the file system e.g. to the directory `cli`:

```
$ pwd
/mnt/galaxy/home/researcher2/thesis
$ cd ../
$ mkdir cli
$ mv thesis/quotes.txt cli/
```

Copy `quotes.txt` to `quotation.txt` using the command `cp`. `cp` works very much like `mv`, except it copies a file instead of moving it:

```
$ cd cli/
$ cp quotes.txt quotations.txt
$ ls
quotes.txt quotations.txt
```

Warning: By default the `cp` and `mv` commands will overwrite a destination file if it already exists.

This means you will lose the old contents of the destination file!

2.2.4 Removing files

Let's clean up and remove the file `quotations.txt` using the command `rm`

```
$ rm quotations.txt
$ ls
quotes.txt
```

Deleting is Forever! Unix does NOT have a trash bin.

2.3 Using wildcards

`*` is a wildcard. It matches zero or more characters, so `quo*` matches anything starting with `quo`

```
$ ls quo*
quote.txt quotation.txt
```

> `?` is also a wildcard, but it only matches a single character.

2.4 Some other useful tips

- Press `Tab` to autocomplete (useful to avoid errors).
- Re-run last command: `!!`

- List previous commands: `history`
  - `!(command number)` to re-run a command e.g. `!21`
- Up and down arrows to scroll through previous commands -> edit command line.
- Go directly to your home directory: `cd`
- `Ctrl+C` to terminate a command.
- Clear terminal window: `clear`

2.5 Useful resources for beginers

- Rescued by Code!
- The Unix Shell (Software Carpentry materials)
- Stackoverflow

3

PacBio assembly with command-line tools

3.1 Overview

See Figure 3.1.

3.2 Get data

The read files which we will be using in today's practicals are placed in a specific path, independent of your working directory.

We will define path variables for these datasets and use them in our commands.

PacBio

- Define the path to PacBio input fastq files

```
$ pacbio_reads_path=~/workshop_data/Klebsiella_pneumoniae_25727/data/pacbio
```

- List the PacBio read datasets

```
$ ls -lt $pacbio_reads_path/*.subreads.fastq.gz
```

- Concatenate all PacBio read files into a single file. Note that you don't need to run this as this is a preparation step to run Canu.

```
$ cat $pacbio_reads_path/*.subreads.fastq.gz > $pacbio_reads_path/pacbio.fastq.gz
```

`pacbio.fastq.gz`: the input PacBio reads file which will be used for assembly

Illumina

- Define the path to Illumina (short-read) fastq files:

```
$ illumina_reads_path=~/workshop_data/Klebsiella_pneumoniae_25727/data/illumina
```

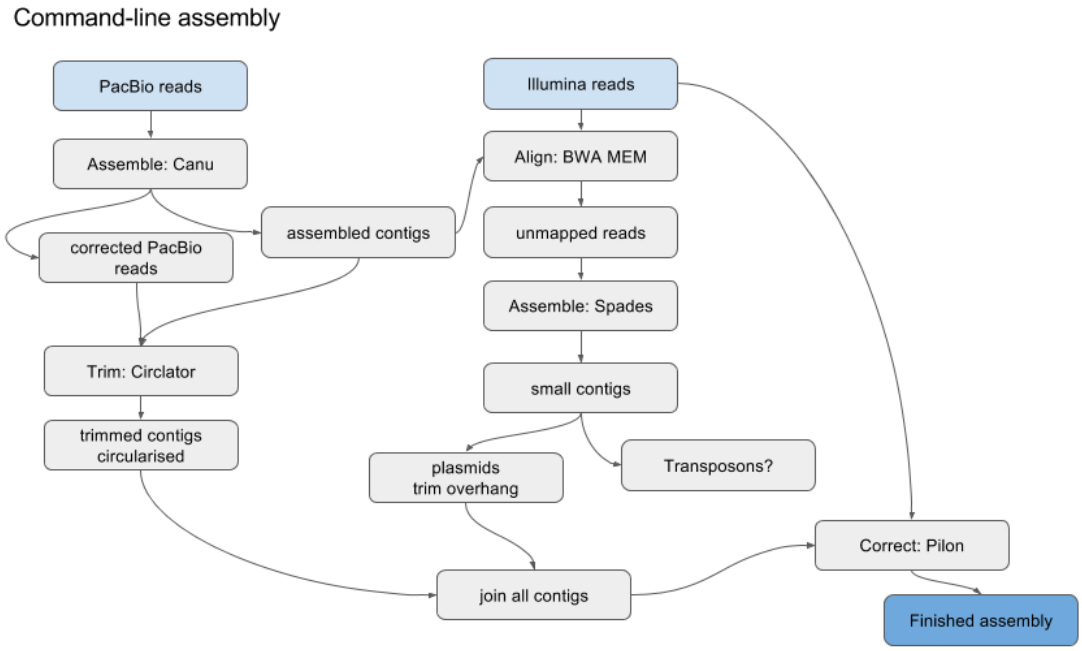

Figure 3.1: Simplified version of the workflow.

- `illumina_R1.fastq`: the Illumina forward reads
- `illumina_R2.fastq`: the Illumina reverse reads

3.3 Assemble

We will use the assembly software called Canu (Koren et al. 2017).

IMPORTANT: since Canu assembly takes around 60 min to excecute in this environment, we have already assembled the genome and provided the pre-assembled genome file in the folder `~/workshop_data/Klebsiella_pneumoniae_25727/canu_outdir`.

Please **DO NOT** run the assembly on the VM instance or you will end up rewriting the folder `canu_outdir` and its contents.

Run Canu with these commands:

```
$ canu -p canu -d canu_outdir \
    genomeSize=5.5m \
    -pacbio-raw $pacbio_reads_path/pacbio.fastq.gz
```

The parameters provided to the program excecutable 'canu' are as follows:

- `-p canu` names prefix for output files (`canu`)
- `-d canu_outdir` names output directory (`canu_outdir`)

- `genomeSize` only has to be approximate; e.g. *Klebsiella pneumoniae*, 5.5 Mbp.
- Canu will correct, trim and assemble the reads.
- Various output statements will be displayed on the screen.

3.3.1 Check the output files created by Canu

Before continuing let's copy the pre-processed data into our `working_dir` folder to avoid problems:

```
$ kp_analysis=~/workshop_data/Klebsiella_pneumoniae_25727/analysis
$ cd ~/working_dir
$ mkdir canu_outdir
$ cp $kp_analysis/canu_outdir/*.* canu_outdir/
```

> With `*.*` we'll copy only the files and not folders from the pre-processed data folder to avoid copying stuff that we won't need, and to save space.

Move into `canu_outdir` and use `ls` command to see the output files.

- `canu.contigs.fasta` holds the assembled sequences.
- `canu.unassembled.fasta` contains the reads that could not be assembled.
- `canu.correctedReads.fasta.gz` contains the corrected PacBio reads that were used in the assembly.
- `canu.contigs.gfa` is the graph of the assembly.
- `canu.report` includes a series of status messages, execution logs, and analyses that are printed on the screen as Canu runs.

3.3.2 Display summary information about the contigs

This will show the contigs found by Canu:

```
$ cd ~/working_dir
$ infoseq canu_outdir/canu.contigs.fasta | less -S
```

Most of the bacterial sized genomes are assembled as single contigs using PacBio long-reads.

Here canu has assembled the PacBio reads into 3 contigs.

The biggest contig is approximately 5.5 million bases, which is around the known assembled genome size of this species.

What are the other two contigs?

- First possibility is that Canu may not have been able to join all the reads into one contig.
- The other possiblity is that the sample may contain some plasmids and these may be found completely or partially assembled by Canu as additional contigs (here two contigs of size ~33 kbp have been assembled).

This will be investigated further in a separate session using tools such as Bandage (Wick et al. 2015).

3.3.3 Change Canu parameters if required

If the assembly is poor with many contigs, there is always an option to re-run Canu with extra sensitivity parameters; e.g.:

```
$ canu -p prefix -d outdir \
    corMhapSensitivity=high corMinCoverage=0 genomeSize=5.5m \
    -pacbio-raw pacbio.fastq.gz
```

> Please use `canu --help` to see all parameters which can be used with Canu.

3.3.4 Questions

Q1

How do long- and short-read assembly methods differ?

Answer

Many short-read assembly programs use: *de Bruijn* graphs.

Long read assemebly involves a move back towards simpler overlap-layout-consensus methods.

Q2

Where can we find out what the approximate genome size should be for the species being assembled?

Answer

NCBI Genomes - enter species name - click on Genome Assembly and Annotation report - sort table by clicking on the column header Size (Mbp) - look at range of sizes in this column.

Q3

In the assembly output, what are the unassembled reads? Why are they there?

Answer

The file `canu.unassembled.fasta` contains reads and low-coverage contigs which could not be incorporated into the primary assembly. Unassembled sequences are primarily low-coverage sequences spanned by a single read.

Q4

What are the corrected reads? How did canu correct the reads?

Answer

The raw reads get a new consensus sequence and are trimmed to well-supported regions.

A detailed explanation is provided by the Canu paper:

> A full Canu run includes three stages: correction, trimming, and assembly. In all stages, the first step constructs an indexed store of input sequences, generates a k-mer histogram, constructs an indexed store of all-vs-all overlaps, and collates summary statistics. The correction stage selects the best overlaps to use for correction, estimates corrected read lengths, and generates corrected reads.

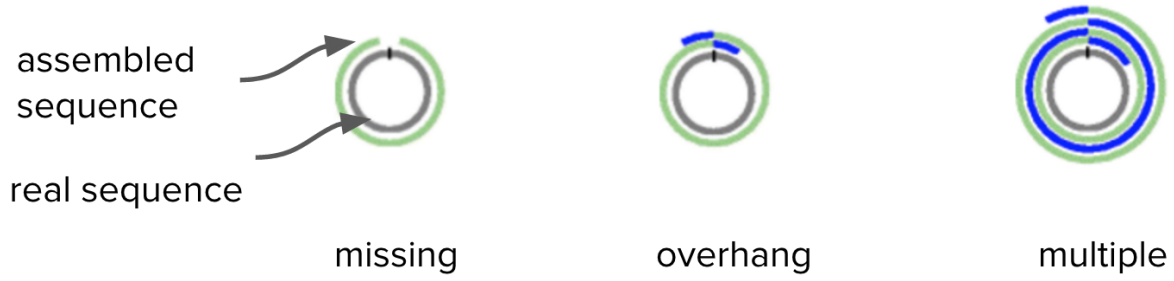

Figure 3.2: Adapted from Figure 1 in Hunt et al. (2015). Overhangs are shown in blue.

Q5

Where could you view the output .gfa and what would it show?

Answer

Bandage: a Bioinformatics Application for Navigating *De novo* Assembly Graphs Easily. Bandage is a program for visualising *de novo* assembly graphs.

By displaying connections which are not present in the contigs file, Bandage opens up new possibilities for analysing de novo assemblies.

3.4 Trim and circularise

3.4.1 Circular nature of molecules

Current long-read assembly software still typically assumes that the contigs they produce are linear. In contrast, the genome of almost every species contains at least one circular DNA structure. Correct completion and circularization of these molecules is essential if they are to be used routinely in clinical practice; e.g.:

- Bacterial chromosomes and plasmids and the plastid: whole-genome sequencing provides improved resolution in bacterial epidemiology and allowis in silico prediction of antimicrobial resistance. Many important antimicrobial resistance and virulence determinants are carried on plasmids, illustrating the importance of having complete and accurate information for these circular sequences.
- Mitochondrial genomes of eukaryotes: in humans, the mitochondrial genome has been implicated in controlling phenotypes such as depression, Leber hereditary optic neuropathy, and myopathy and diabetes mellitus.

3.4.2 Run Circlator

(~19 min runtime)

Circlator (Hunt et al. 2015) identifies and trims overhangs (on chromosomes and plasmids) and orients the start position at an appropriate gene (e.g. *dnaA*).

The program uses the assembled contigs from Canu as well as the corrected reads prepared by Canu as input.

```
$ pwd
/mnt/galaxy/home/researcher2/working_dir

$ circlator all --threads 2 --verbose \
    canu_outdir/canu.contigs.fasta \
    canu_outdir/canu.correctedReads.fasta.gz \
    circlator_outdir
```

- `--threads`: is the number of cores. Change this to an appropriate number, as per your environment.
- `--verbose`: prints additional progress information to the screen.
- `canu_outdir/canu.contigs.fasta` is the file path to the input Canu assembly.
- `canu_outdir/canu.correctedReads.fasta.gz` is the file path to the corrected PacBio reads - note, fasta not fastq.
- `circlator_outdir` is the name of the output directory.

Some output will print to screen. When finished, it should say `Circularized x of x contig(s)`.

Check the output.

Move into the circlator_outdir directory and `ls` to list files.

- Were the contigs circularised?

```
$ less circlator_outdir/04.merge.circularise.log
```

> Yes, the main contig was circularised (see line 3 last column).

```
#Contig      repetitive_deleted  circl_using_nucmer  circl_using_spades  circularised
tig00000002  0                   0                   0                   0
tig00000678  0                   0                   0                   0
tig00000679  0                   1                   0                   1
```

- Where were the contigs oriented (which gene)?

```
$ less circlator_outdir/06.fixstart.log
```

> Look in the "gene_name" column.
>
> The contig has been oriented at **sp|B5XT51|DNAA_KLEP3**, which is another name for *dnaA*. This is typically used as the start of bacterial chromosome sequences.

- What are the trimmed contig sizes?

```
$ infoseq circlator_outdir/06.fixstart.fasta
tig00000001 5444904

Trimmed region size = Original size - trimmed size
                    = 5,469,119 - 5,444,904 (24,215 bases trimmed)
This trimmed part is the overlap.
```

- The trimmed contigs are in the file called `06.fixstart.fasta`.
- Make a copy of this file with the name `contig1.fasta` in a new folder called `identify_plasmids`:

```
$ mkdir identify_plasmids
$ cp circlator_outdir/06.fixstart.fasta identify_plasmids/contig1.fasta
```

Open this file in a text editor, e.g. `nano`:

```
$ nano identify_plasmids/contig1.fasta
```

and change the header to `>chromosome`.

Optional

If all the contigs have not circularised with Circlator, an option is to change the `--b2r_length_cutoff` setting to approximately 2x the average read depth.

3.4.3 Questions

Q1

Were all the contigs circularised? Why/why not?

Answer

- Not all contigs are circular (with overhangs). We will see more about what they are in the Bandage tutorial.

Other possibilities are:

- there are no overhangs in the contigs (or they are way too short). Circlator needs them.
- contigs correspond to non-circular molecules.

Q2

Circlator can set the start of the sequence at a particular gene. Which gene does it use? Is this appropriate for all contigs?

Answer

Uses *dnaA* for the chromosomal contig. For other contigs, uses a centrally-located gene. However, ideally, plasmids would be oriented on a gene such as *repA*. It is possible to provide a file to Circlator to do this.

3.5 Find smaller plasmids

PacBio reads are long, and may have been longer than small plasmids. We will look for any (additonal) small plasmids using the Illumina reads.

This section involves several steps:

1. Use the Canu+Circlator output of a trimmed assembly contig.
2. Map all the Illumina reads against this PacBio-assembled contig.
3. Extract any reads that didn't map and assemble them together: this could be a plasmid, or part of a plasmid.
4. Look for overhang: if found, trim.

3.5.1 Align Illumina reads to the PacBio contig

- Index the contigs file using `bwa index` (H. Li and Durbin 2009; H. Li and Durbin 2010):

```
$ pwd
/mnt/galaxy/home/researcher2/working_dir

$ bwa index identify_plasmids/contig1.fasta
```

- Align Illumina reads using using `bwa mem` (H. Li 2014): *(~4 min)*

```
$ bwa mem -t 2 identify_plasmids/contig1.fasta \
    $illumina_reads_path/illumina_R1.fastq $illumina_reads_path/illumina_R2.fastq \
    | samtools sort > identify_plasmids/aln.bam
```

- `bwa mem` is the alignment tool
- `-t 2` is the number of cores: choose an appropriate number
- `contig1.fasta` is the input assembly file (with the biggest contig which represents the chromosome)
- `illumina_R1.fastq illumina_R2.fastq` are the Illumina reads
- `|` samtools sort pipes the output to samtools to sort
- `>` aln.bam sends the alignment to the file aln.bam

3.5.2 Extract unmapped Illumina reads

- Index the alignment file using SAMtools (H. Li et al. 2009):

```
$ samtools index identify_plasmids/aln.bam
```

- Extract the fastq files from the bam alignment - those reads that were unmapped to the PacBio alignment - and save them in various "unmapped" files:

```
$ samtools fastq -f 2 \
    -1 identify_plasmids/unmapped.R1.fastq \
    -2 identify_plasmids/unmapped.R2.fastq \
    -s identify_plasmids/unmapped.RS.fastq \
    identify_plasmids/aln.bam
```

- `fastq` is a command that coverts a `*.bam` file into fastq format
- `-f 2`: only output unmapped reads
- `-1`: put R1 reads into a file called unmapped.R1.fastq
- `-2`: put R2 reads into a file called unmapped.R2.fastq
- `-s`: put singleton reads into a file called unmapped.RS.fastq
- `aln.bam`: input alignment file

We now have three files of the unmapped reads: `unmapped.R1.fastq`, `unmapped.R2.fastq`, `unmapped.RS.fastq`.

3.5.3 Assemble the unmapped reads

IMPORTANT: since SPAdes assembly would take ~25 min to excecute in this environment, we have already assembled unmapped reads and provided the pre-assembled contigs in the following folder:

```
~/workshop_data/Klebsiella_pneumoniae_25727/identify_plasmids/spades_assembly
```

- Assemble with SPAdes (Bankevich et al. 2012):

```
$ spades.py -1 identify_plasmids/unmapped.R1.fastq \
    -2 identify_plasmids/unmapped.R2.fastq \
    -s identify_plasmids/unmapped.RS.fastq \
    --careful --cov-cutoff auto \
    -o identify_plasmids/spades_assembly
```

> - `-1` is input file forward
> - `-2` is input file reverse
> - `-s` is unpaired
> - `--careful` minimizes mismatches and short indels
> - `--cov-cutoff` auto computes the coverage threshold (rather than the default setting, "off")
> - `-o` is the output directory

- Before continuing let's copy the pre-processed data into our `working_dir` folder to avoid problems:

```
$ mkdir ~/working_dir/identify_plasmids/spades_assembly
$ cd ~/working_dir/identify_plasmids/spades_assembly
$ cp $kp_analysis/identify_plasmids/spades_assembly/*.* ./
```

> With `*.*` we'll copy only the files and not folders from the pre-processed data folder to avoid copying stuff that we won't need, and to save space.

- Move into the output directory (`spades_assembly`) and look at the contigs:

```
$ infoseq contigs.fasta
```

6 contigs were assembled, with the max length of 244 bp.

None of the contigs seem to indicate that they represent a possible plasmid.

However there have been many instances (for other samples) where plasmids missed by PacBio have been assembled by Illumina reads.

3.5.4 Questions

Q1

Why is this section so complicated?

Answer

Finding small plasmids is difficult for many reasons! "On the (im)possibility to reconstruct plasmids from whole-genome short-read sequencing data" is a publication with a nice summary on the subject (Arredondo-Alonso et al. 2017).

Q2

Why can PacBio sequencing miss small plasmids?

Answer

Library preparation includes a size selection step that excludes small fragments. Standard genomic preparation size selects from 15 kbp and above. Therefore molecules smaller than this will be difficult to reconstruct.

Q3

We extract unmapped Illumina reads and assemble these to find small plasmids. What could they be missing?

Answer

Repeats that have mapped to the PacBio assembly.

Q4

How do you find a plasmid in a Bandage graph?

Answer

It is probably circular, matches the size of a known plasmid, and has a *repA* gene.

Q5

Are there easier ways to find plasmids?

Answer

Possibly. One option is the program called Unicycler which may automate many of these steps (Wick et al. 2017).

3.6 Correct/Polish

We will correct the PacBio assembly with Illumina reads. Here we will correct the circularised main chromosome from the previous steps.

We can essentially follow a similar process to 'correct' any other smaller contigs which have been identified.

- Make a folder: `pilon_correction`

```
$ mkdir pilon_correction
```

- Map Illumina reads to the contig(s) which need to be corrected.

> Since we have already done this mapping step when trying to identify "smaller plasmids", we can avoid redoing it.

> Instead we can copy the bam file (Illumina reads aligned to the chromsomal contig) and the Chromosomal contig sequence into the folder `pilon_correction` and index the file for further processing

```
$ cp identify_plasmids/contig1.fasta pilon_correction/
$ samtools faidx pilon_correction/contig1.fasta
```

```
$ cp identify_plasmids/aln.bam pilon_correction/
$ samtools index pilon_correction/aln.bam
```

3.6.1 Run Pilon

(~6 min runtime)

```
$ pilon -Xmx8G --genome pilon_correction/contig1.fasta \
    --frags pilon_correction/aln.bam \
    --output pilon_correction/pilon1 \
    --fix all --mindepth 0.5 --changes --verbose --threads 2
```

- `--genome`: is the name of the input assembly to be corrected
- `--frags`: is the alignment of the reads against the assembly
- `--output`: is the name of the output prefix
- `--fix`: is an option for types of corrections
- `--mindepth`: gives a minimum read depth to use
- `--changes`: produces an output file of the changes made
- `--verbose`: prints additional information to the screen during the run
- `--threads`: set this to an appropriate number

Look at the changes file:

```
$ less pilon1.changes
chromosome:7913 chromosome_pilon:7913 . G
chromosome:19204 chromosome_pilon:19205 . G
chromosome:44344 chromosome_pilon:44346 . G
chromosome:119617 chromosome_pilon:119620 . G
chromosome:154840 chromosome_pilon:154844 . G
chromosome:158873 chromosome_pilon:158878 . C
chromosome:160555 chromosome_pilon:160561 . G
chromosome:218956 chromosome_pilon:218963 . G
chromosome:240016 chromosome_pilon:240024 . G
chromosome:256012 chromosome_pilon:256021 . C
chromosome:427207 chromosome_pilon:427217 . G
chromosome:439888 chromosome_pilon:439899 . G
chromosome:450693 chromosome_pilon:450705 . G
chromosome:456936 chromosome_pilon:456949 . C
chromosome:469780 chromosome_pilon:469794 . G
```

There are 214 changes by pilon to the assembly with most of the changes being single base pair insertions.

Optional

If there are many changes, run Pilon (Walker et al. 2014) again, using the `pilon1.fasta` file as the input assembly, and the Illumina reads to correct.

3.6.2 Questions

Q1

Why don't we correct earlier in the assembly process?

Answer

We need to circularise the contigs and trim overhangs first.

Q2

Why can we use some reads (Illumina) to correct other reads (PacBio) ?

Answer

Illumina reads have higher accuracy.

Q3

Could we just use PacBio reads to assemble the genome?

Answer

Yes, if accuracy is adequate.

4

Visualisation tools

To open the VM's Graphical User Interface (GUI), you will need to access via your web browser, either Firefox or Chrome.

- Write your assigned IP in the address bar, followed by `/vnc`, e.g. `123.45.67.789/vnc`.
- Now just login with your assigned username and the password.

4.1 Bandage

Bandage (a Bioinformatics Application for Navigating *De novo* Assembly Graphs Easily), is a program that creates interactive visualisations of assembly graphs. Sequence assembler programs, such as Velvet (Zerbino and Birney 2008), SPAdes (Bankevich et al. 2012), Trinity (Grabherr et al. 2011) and MEGAHIT (D. Li et al. 2016), carry out assembly by building a graph, from which contigs are generated. By granting easy access to these assembly graphs, Bandage allows users to better understand, troubleshoot and improve their assemblies.

With Bandage, you can zoom and pan around the graph, customise the visualisation, search for sequences, extract sequences, and more.

4.1.1 Basics

To run Bandage, you need a to open your VM with a GUI. There, open the terminal or select `Run` on the Linux main menu and type:

```
$ Bandage
```

To navigate around Bandage, you can use these controls to make your life easier:

- `Ctrl+Right Mouse Button`: panning.
- `Ctrl+Left Mouse Button`: rotate view.
- `Ctrl+Mouse Wheel (up/down)`: zoom (in/out).

Note: in Mac, use the `Cmd` key instead of `Ctrl`.

In Bandage, there are two main types of data: nodes and edges. Without getting into the details of how assemblies are generated, nodes refer to sequences while the edges refer to possible paths that connect the nodes.

Although we are not using the command line for this section, we will use the notation of variables to simplify the file paths:

```
staph47=~/worshop_data/Staphylococcus_aureus_25747/analysis
staph45=~/worshop_data/Staphylococcus_aureus_25745/analysis
kleb27=~/worshop_data/Klebsiella_pneumoniae_25727/analysis
```

4.1.2 Unitigs vs contigs

Canu generates two different `.gfa` files and their associate `.fasta`: `*.unitigs.gfa` and `*.contigs.gfa`. *Unitigs* are uniquely overlapping sets of reads that assembly without any conflicts or alternate paths. *Contigs* are contiguous sets of unitigs generated by resolving conflicting or alternate paths by the assembler.

Before getting into complex assembly graphs, let's look into a rather simple example.

- Go to `File > Load graph`. Select the following file: `$staph47/canu_outdir/canu.unitigs.gfa`.
- On the left side menu click on `Draw graph`.

4.1.2.1 Questions

4.1.2.1.1 Q1

How many unitigs are present? Can you provide the length and coverage without opening the gfa file in a text editor?

Answer

There are 3 unitigs. To visualise the coverage, name, and size select the appropriate `Node labels` on the left hand menu.

- tig00000001: 35470 bp, 1.00x.
- tig00000002: 2845569 bp, 43.6x.
- tig00000003: 30699 bp, 3.21x.

4.1.2.1.2 Q2

tig00000002 is obviously the bacterial chromosome, but what's the nature/origin of the other unitigs? Do they actually belong to the assembly?

Answer

tig00000001 is a "rogue" single read, but to explore a bit more of why they are there we are going to BLAST all unitigs against each other.

- Click on `Create/view BLAST search` on the left panel.
- Build BLAST database (Camacho et al. 2009; Altschul, Gish, and Miller 1990).
- Load from FASTA file. Select the `$staph47/canu_outdir/canu.unitigs.fasta` file.
- Run BLAST and close the window.
- Now back on the main window change the graph display on the drop-down menu to `BLAST hits (rainbow)`.
- On the left side menu change the `Query` under the `BLAST` section.

As you can see, the short unitigs overlap with the tips of the chromosome at the place where it should close. They are redundant and we could even considered as "assembly rubbish". Although, many of these fragments are curated when Canu creates the contigs, they can still appear in the contig output files. You can see the `$kleb27/canu_outdir/canu.contigs.gfa` file from the *Klebsiella pneumoniae* assembly as example (we will use it in the next exercise).

4.1.2.1.3 Q3

So, can we get rid of them in Bandage?

Answer

Yes.

- First, select the two unitigs we want to remove.
- `Edit > Remove selection from graph`
- Now you can save your gfa and/or fasta files through the `Output` menu.

4.1.2.1.4 Q4

How does this compare to the `*.contigs.gfa` generated by Canu? In a new instance of Bandage open `$staph47/canu_outdir/canu.contigs.gfa`.

Answer

They are exactly the same. This process is what Canu do automatically. However, you often find this "unresolved issues" in the final `*.contigs.gfa` files too, and that's where this comes in handy.

4.1.3 Short- vs long-read assemblies

In this section we evaluate the differences between short-read and long-read assemblies using ***Klebsiella pneumoniae* strain 25727**. .

- Open two instances of Bandage:
  - In one, open the Canu assembly:
    `$kleb27/canu_outdir/canu.contigs.gfa`
  - In the other, open the Spades assembly:
    `$kleb27/spades_illumina_assembly/assembly_graph_with_scaffolds.gfa`
- Draw both gfa.

Aside of the evident contiguity and linearity of the long-read assemblies, the other advantage is how repetitive elements are handled and resolved. A universal example in bacteria and archaea is the ***rRNA gene cluster***. It is often present in multiple copies, usually close to each other or even concatenated. This cluster spans the 16S (~1550 bp), the 23S (~2900 bp) and the 5S (~110 bp) rRNA genes, usually in that order and with a tRNA gene between the 16S and the 23S.

- Create a BLAST database like in the previous exercise for the short-read assembly ONLY for now.
- Load the `$kleb27/visualisation/genomes/kp_rrna_genes.fasta` file as query. This file contains sequences for the 16S, 23S and 5S genes from *Klebsiella pneumoniae*.
- Adjust the BLAST parameters by clicking in `Set BLAST hit filters` and set identity threshold to 90%.
- BLAST!

4.1.3.1 Questions

4.1.3.1.1 Q1

Would you be able to say how many copies of the rRNA gene cluster are there based on the short-read assembly? How? What other problems do you see in the region?

Answer

Yes, but you would only get an estimate.

You can either use differential coverage or assembly paths.

Repetitive regions like the rRNA gene cluster, don't only cause a collapse of the multiple copies into few (often only one), but also can induce fragmentation due to heterogeneity between copies.

4.1.3.1.2 Q2

Could you get the copy number information from the short-reads assembly using the coverage information?

Answer

You can estimate the number of copies with the coverage of the contigs with highest coverage of the rRNA gene cluster (i.e. conserved regions, 177x to 193x) and the coverage of the well-assembled contigs (i.e. long contigs, most between 24x to 29x). If you divide the coverages you get a range between 6.1 to 8.0.

4.1.3.1.3 Q3

Could you get the copy number information from the short-reads assembly using assembly paths?

Answer

Another way to estimate the copy number, more dependent on the complexity of the assembly graph, is based on the paths that leave and come back to the cluster of interest. To do this we redraw the graph based on the BLAST hits:

- On the left side click on the drop-down menu under `Scope` in the `Graph drawing` section and select `Around BLAST hits` and a `Distance` of between 6-8 to simplify the "hairball".
- Now you need to count how many possible "simple" loops or paths that join the ends of the gene cluster in a rather direct manner. You should find at least 5 "evident" paths.

4.1.3.1.4 Q4

Time to use the long-read assembly. How many copies can you find now?

Answer

The long-read assembly shows 8 uninterrupted copies of the gene cluster.

4.1.4 Curating non-chromosomal elements in a genome assembly

In this section we'll work with the Canu assembly of ***Staphylococcus aureus* strain 25745**.

- In Bandage, load `$staph45/canu_outdir/canu.contigs.gfa`.
- Inspect the assembly and run BLAST using the corresponding fasta file as query: `$staph45/canu_outdir/canu.contigs.fasta`

4.1.4.1 Questions

4.1.4.1.1 Q1

The node 1 is obviously the bacterial chromosome. Can you tell what is node 14?

Answer

Node 14 has "similar" coverage to the chromosome and a size in the range of a plasmid. When showing the BLAST hits of this node, the hits against itself look messy based on the rainbow colors. In some instances, this could indicate that it is a group of misassembled reads. Although based on the coverage seems reliable, it's better to double check. For that, we will use the `$staph45/canu_outdir/canu.unitigs.fasta` file as BLAST query.

Select the tig00000004 as query in the drop-down menu. Do you see anything interesting? A repeating pattern. This indicates that node 14 is likely to be circularised in Circlator and real as it matches a unitig. If you BLAST this putative plasmid on NCBI you will see many hits with *Staphylococcus aureus* plasmids.

If you want to dig further into this. This plasmid and another one of ~2.4 kbp were recovered with the Illumina reads. While mid-large size plasmids are recovered more easily now with Canu, small ones might not even be present in the sequencing data due to the library size selection.

4.1.4.1.2 Q2

Now, what about the two linked contigs?

Answer

Nodes 714 and 715 have a size close to the lower limit known for plasmids and a coverage that could indicate its presence in >10 copies per cell. They map perfectly one to another, so they should have collapsed.

What it is unusual is to have high coverage of something that you would expect to be lost on sequencing or even during assembly.

If you BLAST this "thing", the top match is a synthetic construct used during sequencing as internal control. The bad side, is that it is only indicated by the top 2 matches and all others are anotated as plasmids of multiple bacterial species. Remember how PhiX sequences are everywhere in Illumina-based assemblies? Same thing...

4.2 Integrative Genomics Viewer (IGV)

In this section we will explore our genome assembly, and visualise how the reads map to our genome using the Integrative Genomics Viewer (IGV) (Robinson et al. 2011; Thorvaldsdóttir, Robinson, and Mesirov 2013).

N.B. Please note that IGV 2.4 or later is required when working with long-read data.

4.2.1 IGV data

Define the path to the directory with our visualisation data:

```
$ vis_dir=~/workshop_data/Klebsiella_pneumoniae_25727/analysis/visualisation
```

Go to the visualisation directory:

```
$ cd $vis_dir
```

4.2.2 IGV demo and quick introduction

In this demonstration I will show you our read data (long- and short-reads respectively) mapped onto the already existing and publicly available genome assembly *Klebsiella pneumoniae* ASM36438v3. (Later you will use IGV in the same way to explore the *Klebsiella pneumoniae* assembly you made during this workshop.)

I will:

- Launch IGV.
- Adjust IGV preferences.
- Load the ASM36438v3 "reference" genome into IGV:
  - `Genomes > Load Genome From File...` and select:
    `$vis_dir/genomes/Klebsiella_pneumoniae.ASM36438v3.dna.chromosome.1.fa`

- Go to `File > Load From File...` and load:
  - The annotation (GFF3) file:
    `$vis_dir/annotation/Klebsiella_pneumoniae.ASM36438v3.40.chromosome.1.gff3`
  - The (PacBio) long-reads BAM file:
    `$vis_dir/mapping/PacBioReads_vs_Kp_ref_genome.ASM36438v3.mapped.sort.bam`.
  - The (Illumina) short-reads BAM file
    `$vis_dir/mapping/IlluminaReads_vs_Kp_ref_genome.ASM36438v3.mapped.sort.bam`.
  - The mapped SPAdes contigs BAM file:
    `$vis_dir/mapping/SPAdesContigs_vs_Kp_ref_genome.ASM36438v3.mapped.sort.bam`

Zoom into less than 100 kbp for the BAM file tracks to display reads.

Both our PacBio long-reads and our Illumina short-reads from our "sample" from *K. pneumoniae* strain 25727 only map to certain regions of the downloaded *K. pneumoniae* ASM36438v3 reference genome. It thus appear as though our *K. pneumoniae* strain 25727 is quite different from the reference strain ASM36438v3.

However, we can still use this example to understand/learn what different things mean when looking at our read alignments in IGV, and what the different options mean and do:

- Transparent or white reads have mapping quality of zero, which depending on the read mapping algorithm can mean different things, but most often means that the read has mapped to multiple locations equally well, e.g. 1:4,094,195 - 4,108,390.
- Insertions are displayed as purple 'I's.
- Deletions are displayed as black bars '-'.
- Highlight a read in IGV by `Ctrl+Left Mouse Click` (Mac: `Cmd+Click`).
  - This will highlight the selected read and its paired mate in the same colour. (You can go to the mate pair read by choosing the `Go to mate` option in the pop-up window that appears.)
- For pair-end reads:
  - Blue read: shorter than expected insert size.
  - Red read: longer than expected insert size.
  - Inter-chromosomal rearrangements are colour-coded by chromosome, e.g. 1:4,108,215 - 4,122,410.
  - Green, teal and dark blue: inversions, duplications and translocations, e.g. 1:14,644 - 22,560.

For more information on viewing alignments in IGV read here.

4.2.3 Long-read sequencing advantages over short-read sequencing

Examples of where long-read sequencing data shows us something we would not have discovered using only short-read data:

- Long-read data can span longer regions, and thus more complex regions can be resolved, which short-read data cannot.
- Long-read data may have more errors, but they are random, while short-read data is not, and therefore even with high coverage systematic errors might show up as SNPs or insertions or deletions (INDELs).
- Here we have not only mapped long- and short-read data to a reference genome respectively, but because we have enough coverage we also made a *de novo* assembly (our SPAdes assembly), and therefore we were able to map the contigs from this assembly to the reference genome. This seems to be consistent with much of the mapped long-reads, but there are still regions that the SPAdes assembly cannot map.

Moreover, our finished (circulated and polished) Canu assembly, which was made from PacBio long-read data, assembled into a single contig. Meanwhile, our finished SPAdes assembly, which was made from Illumina short-read data consists of 202 contigs.

Let's look at and compare the following two regions on chromosome 1 (your only chromosome):

1. Coordinate window 3,172,175 - 3,190,002 and specifically look at the region at 3,180,511 - 3,181,660

- From the binary sequence alignment map (BAM) file created from the short-read data all we can say is that the short-reads will not map to this region.
- The BAM file created from the long-read data however, shows us we have a deletion of ca x nucleotides.

2. Coordinate window 3,238,210 - 3,247,123 and specifically look at the region at 3,241,470 - 3,242,210
   - From the BAM file created from the short-reads again all we can see is that the reads will not map to this region of the genome
   - The BAM file from the long-reads however, shows us that the region in question is highly variable compared to the reference genome, which is why the short-reads couldn't map.

> NOTE: You will need to indicate the chromosome name followed by the coordinates. Write exactly as this in the coordinates window `1:3172175-3190002` to perform the automatic window relocation to region 3,172,175 - 3,190,002 on chromosome 1.

Thus, we have two regions which look identical in the short-read created BAM file (the short-reads will not map to the reference genome), while they turn out to be very different in the long-read created BAM file (a deletion in the former case and a highly variable region in the latter case).

4.2.4 Student exercises

4.2.4.1 Launch IGV

- Launch IGV
- Adjust IGV preferences (for more details read here):
  - `View > Preferences... > Alignments`
    - Alignment Track Options: `Hide indels < 10 bases`
    - Coverage Track Options: `Coverage allele-fraction threshold: 0.25`
    - Tick: `Quick consensus mode`
    - Press OK

4.2.4.2 Load data

- Load the reference genome - our Canu assembly of *K. pneumoniae* strain 25727:
  - `Genomes > Load Genome From File...`
    `$vis_dir/genomes/Klebsiella_pneumoniae_final_Canu.fasta`
- Load the genome annotation (GenBank file):
  - `File > Load From File...`
    `$vis_dir/annotation/Klebsiella_pneumoniae_final_Canu.gbk`
- Load the long-read BAM file:
  - `File > Load From File...`
    `$vis_dir/mapping/PacBioReads_vs_CanuAssembly.mapped.sort.bam`
- Load the short-read BAM file:
  - `File > Load From File...`
    `$vis_dir/mapping/IlluminaReads_vs_CanuAssembly.mapped.sort.bam`
- Load BAM file made from contigs from SPAdes assembly mapped onto our Canu reference assembly:
  - `File > Load From File...`
    `$vis_dir/mapping/SPAdesContigs_vs_CanuAssembly.mapped.sort.bam`
- Load BAM file made from our unpolished Canu assembly mapped onto our Canu reference assembly:
  - `File > Load From File...`
    `$vis_dir/mapping/UnpolishedCanuAssembly_vs_CanuAssembly.mapped.sort.bam`

4.2.4.3 Inspect the Sequence Alignment Maps

Zoom in to display the reads in the tracks of the BAM files (<100 kbp). Move along genome.

4.2.4.3.1 Q1

What do you see? Can you find big regions that differ between the track from the long-read map and the short-read map respectively? What do these differences mean?

Answer

For example, go to 436,469 - 449,807. It's a region mapping to multiple locations with the short-read data, but uniquely with the long-read data.

> NOTE: You will need to write exactly as this in the coordinates window `chromosome_pilon:436469-449807` to perform the automatic window relocation.

4.2.4.3.2 Q2

How does the read coverage compare between the BAM file made from the long- vs shor-read data respectively?

Answer

The coverage from the long-read data is more uniform and consistently high, while the coverage from the short-read data goes up and down like a roller coaster - in parts dropping almost to zero.

4.2.4.3.3 Q3

Can you identify a region that the short-read created BAM file and the SPAdes assembly identifies as a SNP, while the long-read created BAM file and Canu assembly does not identify as a SNP?

Answer

Coordinate window: 2,957,861 - 2,966,789 with an A > C SNP at 2,962,238 (according to short-read data) but not a SNP at all according to long-read data.

4.2.4.3.4 Q4

Set the coordinate window to 53,579 - 53,619 and look at nucleotide 53,595. What do you see?

Answer

A quick glance at the BAM file from the Illumina short-read data suggests a T > G SNP at nucleotide 53,595. However, at closer inspection (hover with the mouse over the nucleotide), that is not the case (T:61%, G:35%, ref:T). PacBio long-read data does not identify this nucleotide as a SNP.

4.2.4.3.5 Q5

Set the coordinate window to 5,245,060 - 5,245,100 What can you say about nucleotide 5,245,080?

Answer

A quick glance at the BAM file from the Illumina short-read data suggests a T > G SNP at nucleotide 5,245,080. However, at closer inspection (hover with the mouse over the nucleotide), that is not the case (T:58%, G:42%, ref:T). PacBio long-read data does not identify this nucleotide as a SNP.

4.2.4.3.6 Q6

Set the coordinate window to 3,409,910 - 3,481,345 and look at the track displaying the contigs from the unpolished Canu assembly mapped against the final, circularised and polished assembly. What do you see?

Answer

The unpolished assembly suggests there are SNPs in this region. However, there are only two contigs to support this, and the final assembly have removed the SNPs in the polishing step.

4.2.4.4 Inspecting the annotation

If you have been working with IGV locally (on the Windows machine), then log in to the VM using the command-line prompt. However, if you are already logged into the VM through the web browser just open the terminal of the VM.

We are going to look through the Prokka - generated annotation (Seemann 2014) files from our Canu (made from PacBio long-reads) and SPAdes (made from Illumina short-reads) assemblies respectively.

In particular, we are going inspect what the BAsic Rapid Ribosomal RNA Predictor (Barrnap) found. The rRNA gene cluster is often present in multiple copies in bacteria. Therefore, it is very possible that these regions will be collapsed when short-reads are used to generate Sequence Alignment Maps (SAMs), while they will hopefully be resolved by the long-read data.

4.2.4.4.1 Q1

What is the difference in number of detected rRNAs by Barrnap in the Canu and SPAdes assembly respectively?

Answer

```
$ cd $vis_dir/annotation
$ grep barrnap Klebsiella_pneumoniae_final_Canu.gff | less -S
$ grep barrnap Klebsiella_pneumoniae_final_SPAdes.gff | less -S
```

First of all, we get 5 times as many entries when searching the Canu annotation for rRNA genes compared to when we do the same search in the annotated SPAdes assembly.

```
$ grep -c barrnap Klebsiella_pneumoniae_final_Canu.gff
25

$ grep -c barrnap Klebsiella_pneumoniae_final_SPAdes.gff
5
```

Moreover, in the Canu assembly the rRNA gene cluster (with the 16S, 23S and 5S rRNA genes) seems to be properly resolved. Not only do we see the rRNA genes appear several times, but we can also detect an inverted ordering of one of the rRNA gene clusters.

In the SPAdes assembly on the other hand the rRNA gene cluster is collapsed, and not only is the gene cluster collapsed, but the rRNA genes themselves are only partially resolved.

4.2.4.4.2 Q2

What are some of the sizes and alignment quality (percentage aligned) of these genes in the Canu and SPAdes assembly respectively?

Answer

In the Bandage section we learned the approximate sizes of the genes that make up the rRNA gene cluster are: 16S rRNA gene (~1550 bp); 23S rRNA gene (~2900 bp); 5S rRNA gene (~110 bp). The rRNA genes in the

annotation of the Canu assembly seem to be complete, while they are only partial in the annotated SPAdes assembly. Note how it says that the ribosomal RNA is partial e.g., `note=aligned only 63 percent of the 16S ribosomal RNA;product=16S ribosomal RNA (partial)` when checking the SPAdes annotation.

```
$ grep barrnap Klebsiella_pneumoniae_final_SPAdes.gff | less -S
```

4.2.4.4.3 Q3

In IGV zoom into the followig coordinates: 5,145,952 - 5,154,853. What do you see?

Answer

The short-reads are mapping to multiple places. The SPAdes assembly (made from short-reads) cannot resolve this region. The BAM file made from long-reads and the (long-read) Canu assembly have no issues resolving this region of the rRNA genes.

4.2.4.4.4 Q4

Now go to region 5,282,887 - 5,291,815. What do you see?

Answer

Same thing as previous region. It is the rRNA gene cluster again (see Q3).

4.2.4.4.5 Q5

Set the coordinates window to 4,377,228 - 4,377,956, and have a closer look at the SNPs at coordinates:

- 4,377,529. A:69%, G:24%, ref:A.
- 4,377,536. T:71%, C:25%, ref:T.
- 4,377,537. G:72%, A:26%, ref:G.
- 4,377,731. A:74%, T:26%, ref:A.

What can you say about these SNPs?

Answer

The first three SNPs are located in a 23S rRNA gene, which is part of the rRNA gene cluster (mentioned above), while the last SNP is located in the StyR-44 non-coding RNA gene. As the genome have multiple copies of these genes short-reads fail to map uniquely to this region, and hence SNPs cannot be identified in this region. Moreover, Pilon, which uses short-read data for correction will therefore fail to correct this region of the genome.

4.2.4.5 Coverage

In this section we will be comparing the read coverage between our BAM files generated using long- and short-read data respectively. As a starting point I will suggest a few coordinates that you can look through in the questions below.

4.2.4.5.1 Q1

In IGV set the coordinates window to 5,348,690 - 5,357,618. Look at nucleotide 5,353,289. What do you observe?

Answer

Illumina reads drops to only 6x coverage.

4.2.4.5.2 Q2

In IGV set the coordinates window to 5,413,409 - 5,413,966. Look at nucleotide 5,413,688. What is the difference between the long-read and short-read generated BAM files?

Answer

95x coverage with PacBio long-reads, but read coverage drops to only 4x coverage with Illumina short-reads.

4.2.4.5.3 Q3

A region where there are no short-reads mapping onto the reference genome, but where there are long-reads mapping to the region is visible in IGV when the window coordinates are set to `chromosome_pilon:1,798,021-1,802,016`.

Can you find another region like this with these datasets loaded?

Answer

Coordinate window: 1,823,986 - 1,828,449

4.2.5 Usage notes

> **N.B**: You will need to write exactly as this in the coordinates window `chromosome_pilon:5145952-5154853` to perform the automatic window relocation.

> **N.B** The BAM file should be sorted and there should be an index file (`.bam.bai`) in the same directory as the BAM file.

> The annotation was generated using Prokka (Seemann 2014) - a toolkit by Torsten Seeman - with the Barrnap option for ribosomal RNA gene prediction.

References

Altschul, Sf, Warren Gish, and W Miller. 1990. "Basic Local Alignment Search Tool." *J Mol Biol.* 215: 403–10. doi:10.1016/S0022-2836(05)80360-2.

Arredondo-Alonso, Sergio, Rob J Willems, Willem van Schaik, and Anita C Schürch. 2017. "On the (im)possibility of reconstructing plasmids from whole-genome short-read sequencing data." *Microbial Genomics* 3 (10): e000128. doi:10.1099/mgen.0.000128.

Bankevich, Anton, Sergey Nurk, Dmitry Antipov, Alexey a. Gurevich, Mikhail Dvorkin, Alexander S. Kulikov, Valery M. Lesin, et al. 2012. "SPAdes: A New Genome Assembly Algorithm and Its Applications to Single-Cell Sequencing." *Journal of Computational Biology* 19 (5): 455–77. doi:10.1089/cmb.2012.0021.

Camacho, Christiam, George Coulouris, Vahram Avagyan, Ning Ma, Jason Papadopoulos, Kevin Bealer, and Thomas L Madden. 2009. "BLAST+: architecture and applications." *BMC Bioinformatics* 10 (1): 421. doi:10.1186/1471-2105-10-421.

Grabherr, Manfred G, Brian J Haas, Moran Yassour, Joshua Z Levin, Dawn A Thompson, Ido Amit, Xian Adiconis, et al. 2011. "Full-length transcriptome assembly from RNA-Seq data without a reference genome." *Nature Biotechnology* 29 (7): 644–52. doi:10.1038/nbt.1883.

Hunt, Martin, Nishadi De Silva, Thomas D. Otto, Julian Parkhill, Jacqueline A. Keane, and Simon R. Harris. 2015. "Circlator: Automated Circularization of Genome Assemblies Using Long Sequencing Reads." *Genome Biology* 16 (1): 294. doi:10.1186/s13059-015-0849-0.

Koren, Sergey, Brian P Walenz, Konstantin Berlin, Jason R Miller, Nicholas H Bergman, and Adam M Phillippy. 2017. "Canu: scalable and accurate long-read assembly via adaptivek-mer weighting and repeat separation." *Genome Research* 27 (5): 722–36. doi:10.1101/gr.215087.116.

Li, Dinghua, Ruibang Luo, Chi-Man Liu, Chi-Ming Leung, Hing-Fung Ting, Kunihiko Sadakane, Hiroshi Yamashita, and Tak-Wah Lam. 2016. "MEGAHIT v1.0: A fast and scalable metagenome assembler driven by advanced methodologies and community practices." *Methods (San Diego, Calif.)* 102 (June): 3–11. doi:10.1016/j.ymeth.2016.02.020.

Li, Heng. 2014. "Toward better understanding of artifacts in variant calling from high-coverage samples." *Bioinformatics (Oxford, England)* 30 (20): 2843–51. doi:10.1093/bioinformatics/btu356.

Li, Heng, and Richard Durbin. 2009. "Fast and accurate short read alignment with Burrows-Wheeler transform." *Bioinformatics* 25 (14): 1754–60. doi:10.1093/bioinformatics/btp324.

———. 2010. "Fast and accurate long-read alignment with Burrows-Wheeler transform." *Bioinformatics* 26 (5): 589–95. doi:10.1093/bioinformatics/btp698.

Li, Heng, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, and Richard Durbin. 2009. "The Sequence Alignment/Map format and SAMtools." *Bioinformatics (Oxford, England)* 25 (16): 2078–9. doi:10.1093/bioinformatics/btp352.

Robinson, James T, Helga Thorvaldsdóttir, Wendy Winckler, Mitchell Guttman, Eric S Lander, Gad Getz, and Jill P Mesirov. 2011. "Integrative genomics viewer." *Nature Biotechnology* 29 (1): 24–26.

doi:10.1038/nbt.1754.

Seemann, T. 2014. "Prokka: rapid prokaryotic genome annotation." *Bioinformatics* 30 (14): 2068–9. doi:10.1093/bioinformatics/btu153.

Thorvaldsdóttir, Helga, James T Robinson, and Jill P Mesirov. 2013. "Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration." *Briefings in Bioinformatics* 14 (2): 178–92. doi:10.1093/bib/bbs017.

Walker, Bruce J, Thomas Abeel, Terrance Shea, Margaret Priest, Amr Abouelliel, Sharadha Sakthikumar, Christina A Cuomo, et al. 2014. "Pilon: an integrated tool for comprehensive microbial variant detection and genome assembly improvement." Edited by Junwen Wang. *PloS One* 9 (11): e112963. doi:10.1371/journal.pone.0112963.

Wick, Ryan R, Louise M Judd, Claire L Gorrie, and Kathryn E Holt. 2017. "Unicycler: Resolving bacterial genome assemblies from short and long sequencing reads." Edited by Adam M. Phillippy. *PLoS Computational Biology* 13 (6): e1005595. doi:10.1371/journal.pcbi.1005595.

Wick, Ryan R, Mark B Schultz, Justin Zobel, and Kathryn E Holt. 2015. "Bandage: interactive visualization of de novo genome assemblies." *Bioinformatics (Oxford, England)* 31 (20). Oxford University Press: 3350–2. doi:10.1093/bioinformatics/btv383.

Zerbino, Daniel R, and Ewan Birney. 2008. "Velvet: algorithms for de novo short read assembly using de Bruijn graphs." *Genome Research* 18 (5): 821–9. doi:10.1101/gr.074492.107.