Long-Read Data Analysis Workshop

Bioinformatics - Nanopore Sequence analysis

27th-28th September 2018
Ramaciotti Center for Genomics

Tim Kahlke
tim.kahlke@uts.edu.au
Github: https://github.com/timkahlke
Twitter: @AdvancedTwigTec

UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Working environment

Login details

Virtual machine

All the tools and data will be provided on the same Virtual machine as yesterday. Login details should have been provided by the workshop organisers.

Course data

The course data for the Nanopore practicals can be found in the workshop data directory in sub-directory *nanopore_practicals.* It includes 2 sub-directories:

- work_dir: Use this directory to work through the different practicals
- data: This directory includes two sub-directories:
    - MinKNOW: a (custom) MinKNOW output directory including fast5 files of a previously run experiment
    - precomp: directory with precompiled data files needed for some of the practicals.

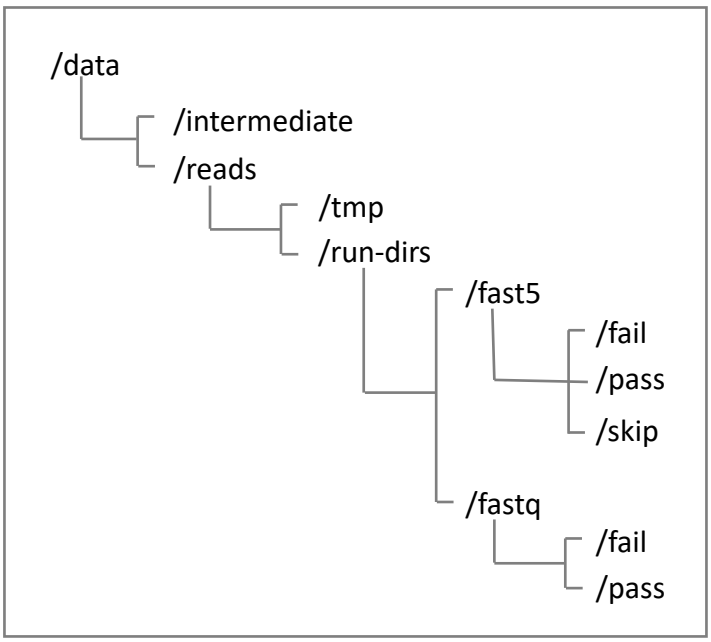
Practical 1
Base calling with Albacore

Find your data

After a (successful) nanopore sequencing run the MinKNOW software will store your data in a specific directory structure.

The nanopore directory includes a “fake” MinKNOW output directory with example data in *fast5* format that will be used throughout the course.

MinKNOW directory structure



Because the provided directory did not include direct base-calling the *fastq* sub-directory is missing. Instead, the raw course fast5 data will be provided in the sub-directory *fast5/skip* of the run directory *20180419_0303.*

Note: MinKNOW labels run-directories with a date prefix (yyyymmdd) followed by an underscore and a specific identifier. Hence, the data provided was sequenced on the 19th of April this year.

Base calling with Albacore

The first step for each sequencing run is to *basecall* your data, i.e., produce fastq sequencing files from the initial fast5 files. We will use the standard basecaller is *albacore* developed by Oxford Nanopore.

Note

Most tools that will be used in this practical are install in a *conda* environment. Before you can use them you will have to activate this environment. This has to be done only once per login to the virtual machine. To activate it type

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

To run albacore and basecall your data first change into the directory WORKSHOP_DIR/*nanopore_practicals/workdir/prac1* where *WORKSHOP_DIR* is the directory in your home directory on the virtual machine that includes *nanopore_practicals*. Then call albacore

```
read_fast5_basecaller.py –i \
WORKSHOP_DIR/nanopore_practicals/precomp/MinKNOW/data/reads/20180419_0303/fast5/skip/0/ \
-s ./albacore_out –f FLO-MIN106 -k SQK-LSK108 -o fastq -t 2
```

NOTE: The backslash "\" in the commands can be ignored throughout the tutorial if you type the command in one continuous line!

The command above will call albacore on the input fast5 directory option (-i), write the output to the directory given with option -s in fastq format (option -o). The options -f and -k defined the flow cell chemistry and the extraction kit, respectively. To see all possible attributes for those two parameters *call read_fast5_basecaller.py -i.*

Base calling with Albacore

While albacore is running you can see the estimated run-time in the lower right corner. For this particular data set it would take several hours to complete the base call. You can use the option *-t NUM* to specify the number of CPUs/cores to speed up the process. However, to save time we already ran alabacore on the data set. You can find the pre-computed data in the directory */nanopore_practicals/data/precomp/albacore_out.* Stop the albacore run by pressing *CTRL-C, r*emove the *albacore_out* directory that was just created by albacore and copy the precompiled albacore output directory into your current directory

```
rm –rf ./albacore_out
cp –R WORKSHOP_DIR/nanopore_practicals/data/precomp/albacore_out .
```

Note: Remember to substitute *WORKSHOP_DIR* with the correct path of the directory with the workshop data in it!

Now and change into the albacore_out directory.
Use *ls* to list the albacore output files and directories:

Configuration.cfg	The configuration file for the base call
Pipeline.log	A log file with base calling information per read
Sequencing_summary.txt	A summary of the sequencing run
workspace	A directory containing the fail and pass fastq directories for the base called reads

You will always find these files in your albacore output directory.

Practical 2

Data analysis

&

Quality control

Data analysis and QC

After the basecalling step you want to get an overview of your data, e.g. what is the number of reads, average length, quality etc.

An increasing number of tools is available to analyse nanopore data. In this tutorial we will use two of the most common tools: *nanoplot* and *poretools*.

Plot your base called reads with nanoplot

Nanoplot is a command-line tool written in python that can be used to visualise your nanopore reads and read alignments to a reference.

Analyse the basecalled data by first find the directory that contains the fastq files that passed the quality control of the MinKNOW base call.

Change into directory */WORKSHOP*_DIR/*nanopore_practicals/workdir/prac2* and create a directory to store the nanoplot output in, e.g. *prac2_nanoplot1.* Then use nanoplot to analyse the fastq files that passed albacores quality threshold:

```
cd WORKSHOP_DIR/nanopore_practicals/workdir/prac2
mkdir prac2_nanoplot1
NanoPlot --fastq ../prac1/albacore_out/workspace/pass \
-o prac2_nanoplot2 --title Passed_reads --loglength
```

The options *--fastq* and *-o* set the input and output for the command. We also want the length of the sequences shown log-transformed (--loglength) and also set a title for the plots (--title).

Data analysis and QC

The above command will create several plots and summary files including a *.html file that can be opened in your browser.

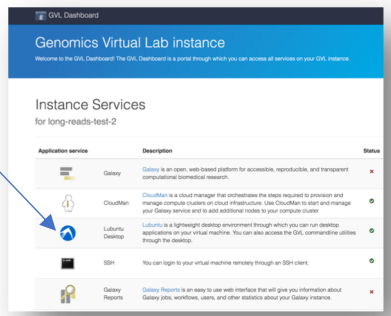
Note

To open NanoPlot result files on the VM you'll have to log on to the VM using a web browser and your login details.

Example:

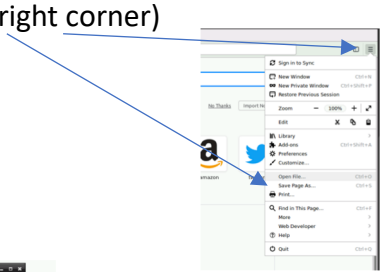
In case you logged in to your VM with the IP address *researcher@12.123.123.12*

1. Open a web browser, e.g. Firefox and go to http://12.123.123.12
2. Choose "Lubuntu" and login with your login details

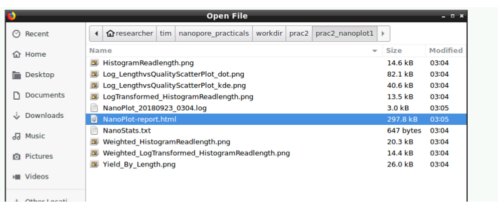


3. Open the browser (lower left corner, you might have to scroll down)

4. Choose *Open File* from the drop-down menue (upper-right corner)



5. Choose file *NanoPlot-report.html*



Data analysis and QC

Inspect the different plots and statistics:

- How many reads do you have in total?
- What is the average, minimum and maximum read length, what is the N50?
- What do the mean quality and the quality distribution of the run look like?

Do the same with the fastq files in the *fail* directory of the albacore output. First create another directory *prac_nanoplot2* and call NanoPlot again:

```
mkdir prac2_nanoplot2
NanoPlot --fastq ../prac1/albacore_out/workspace/fail \
-o prac2_nanoplot2 --title Failed_reads --loglength
```

Open the different plots and compare them.

What are the main differences, e.g., wrt. sequence length and quality?

Does albacore have a default quality cut-off? Which one?

Plotting with poretools

Poretools was one of the first tools to work with nanopore data which provides some plotting functions as well as other functionality to work with fast5 files.

Note:

To work with poretools on the VM you will have to activate a separate conda environment. Also, as soon as you are done with using poretools you will have to deactivate the poretools environment

Data analysis and QC

Activate poretools

```
source activate poretools
```

Deactivate poretools

```
source deactivate poretools
```

Plot the yield of a sequencing run

way of analysing your sequencing run is plotting the read yield of your flow cell over time. Low yield or a drop off of sequencing yield can indicate contaminations in the library or problems with the flow cell itself.

Use the tool *poretools* and the fast5 files to create a yield plot

```
poretools yield_plot \
WORKSHOP_DIR/nanopore_practicals/precomp/MinKNOW/data/reads/20180419_0303/fast5/skip/0/
```

How did the flow cell of the particular run perform?

You will have to close the plot to exit the command

Export fast5 to fasta

Another functionality provided by *poretools* is the export of fasta files from basecalled fast5 files. Use *poretools* to extract fasta files from all fast5 files in */fast5/pass/0*. Use the option --min-length to only extract reads that are at least 2,000 nucleotides long. Because poretools prints the sequences to the terminal you need to *redirect* the output into a file using the ">" operator.

```
poretools fasta --min-length 2000 \
WORKSHOP_DIR/nanopore_practicals/precomp/MinKNOW/data/reads/20180419_0303/fast5/skip/0/ > reads_bigger_2000.fasta
```

You should now have a fasta file called *reads_bigger_2000.fasta* in your current directory.

What did you sequence?

If you are sequencing your own projects you should have a pretty good understanding of where your DNA comes from. However, so far you don't know organism the data for this practical comes from.

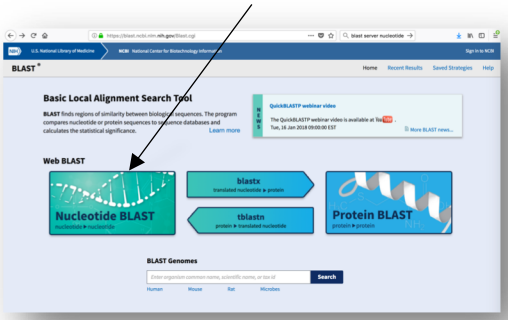
Use the linux command *head* to list the first lines of the fasta file you just created and compare it to known genomes using the *Basic Local Alignment Tool* BLAST at the *Center for Biotechnology Information* (*https://blast.ncbi.nlm.nih.gov/Blast.cgi)*

Data analysis and QC

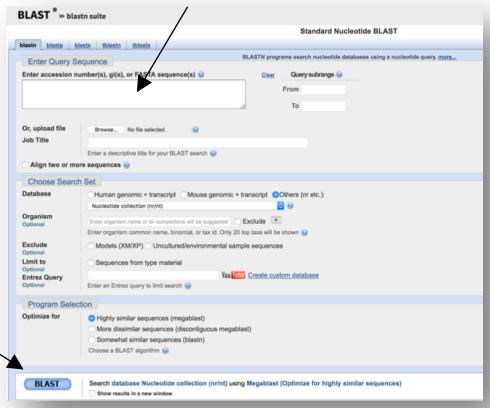
1. Mark and copy sequence (right-click -> copy)

```
tim$ poretools fasta --min-length 2000 /Library/MinKNOW/data/reads/20180117_0306_/fast5/pass/0/ > day1.fasta
tim$ head day1.fasta
bed25f9c7 runid=f7a2672b3a5e8d3654a52e274d2ebdf2d02d4c2c read=1002 ch=202 start_time=2018-01-17T03:20:16Z /Library/MinKNOW/data/reads/20180117_0306_/fast5/pass/0//Tims_Ma
4356_sequencing_run_87673_read_1002_ch_202_strand.fast5
ACATTGTATTGCTGTTAGTTAGTATTCCTATTTGCCCCAGTATTCCAACACTTATTGGGAAATACTCTATTTAATAAAAAAAAATTCTCATGAAACCGTAGCGTATACTTTTGGCGTGAGCAACACAGGGGACAACGTGAAAACACGGCCAATTTATCTCGAATAGGACC
ATAACAAGGTGTCAATTATTCATACATTTCATTGGACACAGTGCCAAACAACAATCCTCCTCAGATTGATCCTTTATGAAGTCTAGGTTCGAGGTGAACCGACACAGTCATATATTTCCTCTTCTGTCTCATGTGTTGTTGCTCGAAGTCACGAGAATACAAAGGTAAGAT
GCAAGTCAATGCGGTGGCGACAGGCGTTCGGCTCGTACATCATCTCGGCGACTTTCGGTTCAGTTGAGTTTCATTCAGTTCGTTTTCGTACGTACCTCTAACTTCTTCGCACCCCATCTCCACACTCACAGACTGAACGCATCAACACCACATAACATCCGACAACGCGA
CTACCTCGTTTACTGCGTTTTCTCCCACAATTATGCTCAAATCCACCTATTTTGCTTCTGCAGCCGCCGTCAGACGTACATTTGCATCCACAGCCACTTCCAACGGCAGCAACTGTGCACGACGACCTGCTGCTGCTGCTTCTGCTTCTACAATTCGCACCATGGCATCG
CCACCCTGCTCACATTCGAAGAACAATCGCCCTCCTCCTACAAATCCCTCCTCTCTCAAACTGCGCACCGTAACCACATCCAACGCGTTACCGTGCTTTCAATACGATCGTCAGATGTTTATGCCACAGCGAAGGAAGAGTTCATTGAACAGGAAGCAATTGGCGGTGCTTGCT
GTGAAGATTGGAATTTTGTTGGAGAAGGCTTTGAAAGATTGGGAGGAGGAGGTAGTTACAAAACAACAGATGTGGAGGTTGATGCCAAACATTTCTTAGCTTGATGACAGCAGCAGCCAAACGTATCTTGGGTTGGAAGCGATTCCTACAAAAAAAGGCAACGTGCATTTCCA
GAAACCTCGCATTACTCTGGTGGGTAAAGGCTACGTGAAACCAACGATTTTGCAACGTTTGCTCCTGCGTTGAAGGATTGTTTGATACTGCCAAGGAAGTGCCAATCTGCAAGAGGCGGGGATAATACTATTTCATTGTACTCGCAAATGTTGGATGAGTTTGAGATGGG
TCGGTCAAGTTCAGAAGGCTTTGGTTCCACTTATCGCAACGATTGGGGAGAGTGAGAACCAACCCAGGATTGAGCACTCTCGGGCTCCGATTGAAGCACAAAGGGACCTGTCGTGAGATTGTGACGGCTATTGGATTTGATGAGACTCATGGGCGGATTGACGTGGTGTT
GATGTGTGAGAATTACAAGTAGATTCACAGAGACGGGTGGTTTCAAGGATTGATGGGTACAATCCACGAGAAGGTGGTCATGCGATGTACGAGCAGAATCTTCGAGAGGCGAGAGTGATTTGAGCTTGGACTCGGCACTTAGTATGGGTGTGCATGAAGCCAAGAGTTTG
GGCTTTCTACAGGTGGGCTGGTCTATTTTGGGAGGCATTTGGGAAGGATGGGAGAAGTTTGACTATTCTGCTGAGGAACTGTAATGCTGCTGTGAATGCGGTTGATTTTGGGAACTTCATTAGAGTGGATGCCGACGAGCTCACGGCTATCCCATACATGTTATCTTGAG
CTGGAAAGTTGGACGTGAACGATGTACCAACTCGATGGAATGCGGAGATGAAGAGCCTTCTCGGCATTGACGTGCTGATGATGCTAATAGTTTGTCTTCAGGATATCCACTGGTCCTGTCTTACGATTGGTTACTTCTGGCGTACCTCCTTAGCTCAATGATGGCTGCTC
CCAGCATGGACAGAAATGATCAAAGCTGGAGAATTCGGGGAGATCAAGAGTGGCTTACGAAGAAAGTGCGTGCATGGAAGGAGGTACAAGTCGTTGGATGATTTGCTCGTGGCTGAAGTGGAGAACCTCTCAACTCACCCAGTATTTCATTGACTATCTGACTGAAAAGT
TCAAAGTATGGCGCAGTAGTAGATATGTGAAGGTTCTTCGTAACTATGAATACCTATGGTATCGTAATTACCTAAACCCTCTATATCTATATCCATGGAACCCATCCGTGGTATATTTCATAGATTGTCTGGATCGCGTTATAGAGTAATCAGCAGTTTGACTTGTCCCG
CCTCTTCGTATCCTTGAAGCTTCTTCGTGTATCTTTGCAATGGCTCTGTCTGCTGCATCATTCCTGTTCGTCTACGACCAGTGACTCCCAGTTCCAGGCACCTCTACTCAGTGCGTCGTCTTCGTTTCGTTGACACTGACGTGCCTTTGCCGGAGTGATGTTCCATTTGTA
ATAGTCAGTAGGTATCTGCATTCGTCCGGAGAACGTGGTAGCTTCTGCTGACTTGGAGAAACTCCTTCTGTGCCGAAGGCCCCATGCCATCGACGACATCGCAGTGGGTCTAGAAAGGAACTGTTTCAGGTGTAGTCAAGATCTAGCTATCGTTGATTGCAAAACGGTTT
CGAAACACACTTCAACGTCGACTCACCGTCCCTTGCTCGACACAATGCCAAAATCAATCTGTACCATTTCACCAGTCTTGGTATGCGCCAATGGGTTACTGAGTATGGCGTCTCCGATACCAAGAATGTCCAACTGAACGGTAGCACGGAGTTAGAAATGGTTGGAACTT
CCAATGCTGTTCACTGCAAGATTTTGTGTACAACGTTCGAGCGGTGTGCCAAAGCCTCCATCGACGTTGAAAAGTATTCAGGAAGAAAAACGGAATACTACGGTAGAAATGGAGGCAACGTTAGACGCCGAAGATTGCACAAGAGATAGCAAACGACCCCGGCAATCGCA
```

2. Select *Nucleotide BLAST*



3. Paste your sequence



4. Start BLAST

Practical 3
Read trimming and adapter removal

Read trimming & adapter removal

Adapter removal using porechop

Similar to other 2nd and 3rd generation sequencing platforms MinION library prep kits use ligate adapters to the ends of the DNA. Additionally, some library preparations kits, such as the older 2D kits, use additional adapters to link the two strands of the DNA. Also, in some instances chimeric reads may occur that include adapters in the middle of the sequence.

Porechop is a freely available open-source tool that among others can be used to find and remove adapters from Oxford nanopore reads and remove chimeric reads.

Change into the *prac3* directory in the *WORKSHOP_DIR/nanopore_practicals/workdir* directory and create a directory *porechop.* Use porechop to remove adapters from the albacore reads that passed the QC.

```
cd WORKSHOP_DIR/nanopore_practicals/workdir/prac3
mkdir porechop
porechop –i ../prac1/albacore_out/workspace/pass/
-o ./porechop/porechopped.fastq --discard_middle
```

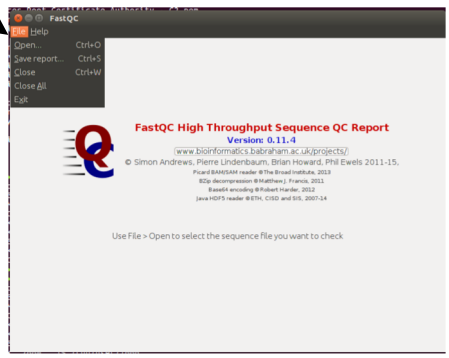
The above command will use the default values of porechop to search for adapters in all fastq files of the input directory, trim the reads and write them to file trimmed.fastq in the created porechop directory. The "--discard_middle" option will remove reads with internal adapters. As mentioned by R. Wick on the porechop github page (https://github.com/rrwick/Porechop), this is essential if you want to use nanopolish later in the analysis.

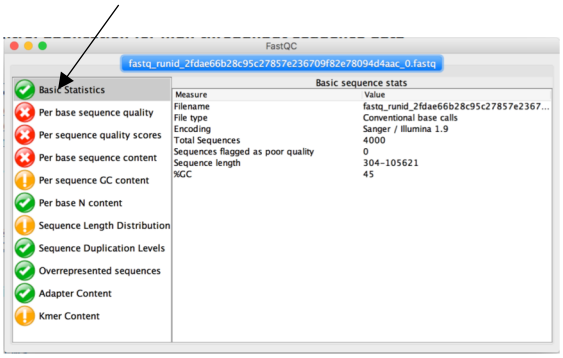
Read trimming & adapter removal

Plot reads using FastQC

In addition to poretools and nanoplot another tool can be used to analyse your sequences visually is FastQC. FastQC is a common tool for quality control of 2nd generation Illumina sequencing reads which can also be use to analyse 3rd generation long-read sequences, e.g. to identify low-quality regions in your read data.

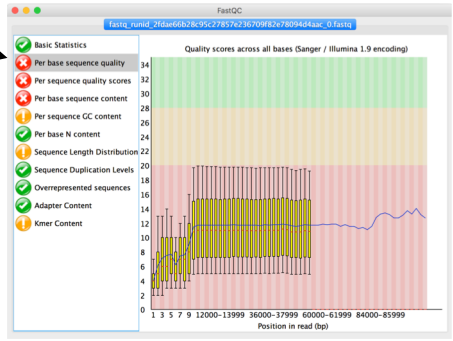
Type "*fastqc –t 2"* on the command-line to open its *graphical user interface* and load the fastq file of the reads that passed albacore





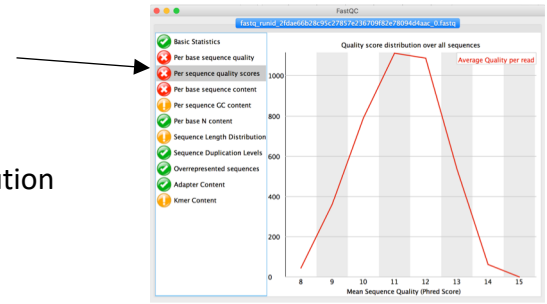
After loading the first tab will show you some basic statistics about your fastq file and sequences

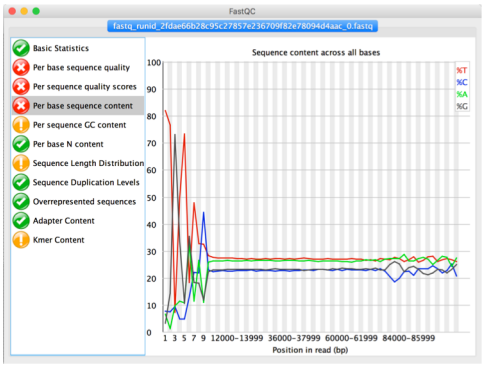
The second tab "Per Base sequence quality" shows the mean and standard deviation of the sequencing quality for each position in all reads of your data set.



Read trimming & adapter removal

The tab “Per Sequence Quality scores” shows the average quality score distribution of your nanopore reads





The tab “Per base sequence contents” show the average ratio of As, Ts, Cs and Gs in your data set. ”Clean” data without sequencing should who almost parallel lines for all four nucleotides.

Using the FastQC output we can make a decision whether we want to trim our reads further, e.g., remove low-quality areas of the reads.

Which areas would you trim off?

Note: Exit FastQC to return to command-line prompt.

Read trimming & adapter removal

Trim reads using NanoFilt

NanoFilt is a python script that can be used to filter reads based on length and quality as well as trimming parts of the sequence.

Create a directory for your nanofilt output in your current working directory (e.g., *WORKSHOP_DIR/nanopore_practicals/workdir/*prac3*/nanofilt*) and then use nanopolish to remove:

- all sequences shorter than 500 nucleotides (option -l)
- trim the first 10 nucleotides off all reads (option --headcrop)

```
mkdir ./nanofilt

NanoFilt –l 500 --headcrop 10 \
 < ./porechop/porechopped.fastq \
 > ./nanofilt/nanofilt_trimmed.fastq
```

NanoFilt does not provide options for input or output files. Therefore we will use the two *redirect* operators “>” and “<“ to

- redirect the file porechopped.fastq into NanoFilt (operator <)
- then redirect the output of NanoFilt into the file nanofilt_trimmed.fastq (>).

Use FastQC to check the result file and compare it to the original fastq.

Practical 4

Assembly using Minimap & miniasm

Assembly using minimap & miniasm

Assembling long-reads, PacBio and Oxford Nanopore, into contiguous sequences (contigs) has been challenging for common 2nd generation assemblers due to the high error rates of 3rd generation sequencing technologies. Recently, an increasing number of assemblers and assembly pipelines is available that take into account he specific characteristics of long-reads.

One way of assembling long-reads is the combination of *minimap* and *miniasm*.
Minimap is read mapper that can identify overlaps in reads.
Miniasm is a very fast overlap assembler that outputs unitigs, i.e., high confidence overlap sequences. In contrast to other long-read assemblers miniasm does not include a consensus step, i.e., miniasm unitigs have a similar error rate as the input reads.

First change intot the prac4 directory and then use minimap to map the filtered nanopore reads onto themselves

```
cd ../prac4
minimap2 –x ava-ont ../prac3/nanofilt/nanofilt_trimmed.fastq \
 ../prac3/nanofilt/nanofilt_trimmed.fastq \
| gzip -1 > minimap.paf.gz
```

Minimap comes with several pre-configured parameter settings depending on the read data. The option *-x ava-ont* sets the default parameters for Oxford Nanopore reads.
The additional *pipe* (|) into *gzip* is useful for large output files to compress them before writing the compressed output file.

Assembly using minimap & miniasm

Use the minimap output and the trimmed reads to assemble unitigs with miniasm.

```
miniasm -f ../prac3/nanofilt/nanofilt_trimmed.fastq \
 ./minimap.paf.gz  > miniasm.gfa
```

Miniamp and miniasm do not provide an option for output files but instead write the output directly to the terminal. Thus, the output has to be redirected using the “>” operator into the result gfa file.

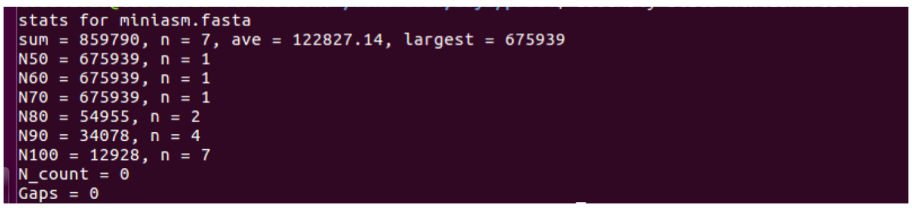
To convert the miniasm.gfa file into a fasta file of unitigs use the following awk command

```
awk ’/^S/{print “>”$2”\n”$3}’ ./miniasm.gfa > miniasm.fasta
```

Now use the tools *assembly-stats* to get some simple statistics about the assembly

```
assembly-stats ./miniasm.fasta
```

Assembly-stats will tell you how many unitigs miniasm assembled, the total length of all



Assembly using minimap & miniasm

How to assess the assembly quality

The best way to analyse the quality of an alignment is to compare it to a published sequence. In directory *WORKSHOP_DIR/nanopore_practicals/precomp/data* you will find the published sequence of chromosome_17 of *Thalassiosira pseudonana.*

First get some assembly statistics. Use *assembly-stats* to print report statistics about this chromosome, e.g., number of nucleotides and number of Ns (gaps or ambiguous sequences).

Compare the two statistics. Do some of the miniasm unitigs match the reference sequence in length?

To make downstream analysis easier first remove sequences that are much shorter than the reference sequence. Use the command *filter_fasta_by_seq_length.pl* from the ampli-tools package to remove all sequences shorter than 100,000 nucleotides from the assembly fasta

```
filter_fasta_by_seq_length.pl –i miniasm.fasta \
-o miniasm_filtered.fasta -a 100000
```

and count the resulting sequences with the linux command *grep*

```
grep –c “>” ~/TOAST2018/day2/prac4/miniasm_filtered.fasta
```

The above command will count the occurance of the character “>” in the input file, which in a fasta file is equivalent to number of sequences.

Assembly using minimap & miniasm

How many sequences “survived” the filtering?

To compare the left unitig to the reference genome use the tool *dnadiff* form the mummer package. Mummer provides fast alignments of large highly similar sequences to each other.

```
dnadiff WORKSHOP_DIR/nanopore_practicals/precomp/data/chr17.fasta \
-p dnadiff  miniasm_filtered.fasta
```

This command will create multiple output files of prefix *dnadiff* with alignment statistics and detailed coordinate information of the overlapping regions of both sequences.

Open the file *dnadiff.report* (e.g. double click) to see a general report of the alignment.
How much of both sequences was aligned?
What is the average percent identity of the alignment?

We can also visualise the alignment with a dot-plot using the command *mummerplot .*

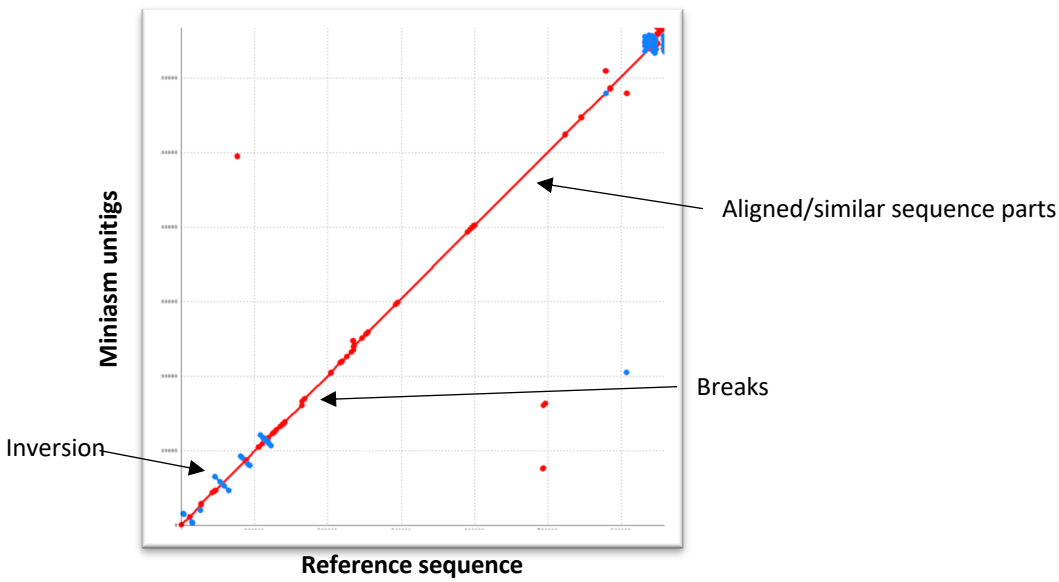
```
mummerplot --png –p miniasm dnadiff.delta \
–R WORKSHOP_DIR/nanopore_practicals/precomp/data/chr17.fasta \
-Q miniasm_filtered.fasta
```

The above command will create an image file in png format in directory prac4. Open it by typing *display miniasm.png*

Assembly using minimap & miniasm

Dot plots

A dot-plot is a visual representation of the similarity of two sequences. In this case the resulting dot-plot shows the miniasm unitig on the y-axis and the reference sequence of chr17 on the x-axis. Sequence parts that overlap are shown as a diagonal line from the lower left to upper right corner. Similarly, orthogonal lines (upper left to lower right) indicate inversions in one of the sequences. Breaks and gaps in the line indicate deletions or insertions in either of the sequences.



What do you think about this initial assembly?

Practical 5
Create consensus sequences using racon

Consensus sequences using racon

Due to consistently high error rates of Oxford Nanopore sequencing data downstream analysis has to include some sort of error correction especially if no high quality short read data is available.
Consensus assemblies try to reduce error rates by choosing the *most likely* sequence of a given assembly and a set of raw reads. Although this does not incorporate the raw signal information of the flow cell to correct individual reads it can significantly improve the quality of an assembly.

The software package racon has been developed to complement the minimap and miniasm tools. It provides a fasta consensus algorithm that uses either 2nd generation short reads or raw noisy long-reads to correct draft assemblies.

To improve a draft assembly with racon map the reads that should be used for error correction against the assembly.
Use minimap to map the trimmed reads from *prac3/nanofilt* against the miniasm assembly and subsequently use the filtered reads and the mapping to build the consensus assembly.

```
cd ../prac5
minimap2 ../prac4/miniasm_filtered.fasta \
../prac3/nanofilt/nanofilt_trimmed.fastq > ./minimap.racon.paf

racon ../prac3/nanofilt/nanofilt_trimmed.fastq \
./minimap.racon.paf ../prac4/miniasm_filtered.fasta \
> ./consensus_assembly.fasta
```

Consensus sequences using racon

Analyse the assembly quality by comparing the consensus assembly to the published sequence using *dnadiff* and mummerplot.

```
dnadiff WORKSHOP_DIR/nanopore_practicals/precomp/data/chr17.fasta \
-p dd_con  ./consensus_assembly.fasta

mummerplot --png –p miniasm ./dd_con.delta \
–R WORKSHOP_DIR/nanopore_practicals/precomp/data/chr17.fasta \
-Q ./consensus_assembly.fasta
```

Did the assembly improve?

Which parts did not improve?

Run racon a second time but this time on the consensus assembly

```
minimap2 ./consensus_assembly.fasta \
../prac3/nanofilt/nanofilt_trimmed.fastq > ./minimap_2.racon.paf

racon ../prac3/nanofilt/nanofilt_trimmed.fastq  \
./minimap_2.racon.paf ./consensus_assembly.fasta \
> ./consensus_assembly2.fas
```

Consensus sequences using racon

Analyse the second consensus

```
dnadiff WORKSHOP_DIR/nanopore_practicals/precomp/data/chr17.fasta \
-p dd_con2  ./consensus_assembly2.fasta

mummerplot --png –p miniasm2 ./dd_con2.delta \
–R WORKSHOP_DIR/nanopore_practicals/precomp/data/chr17.fasta \
-Q ./consensus_assembly2.fasta
```

What are the improvements?

Did some statistics decrease? If so which and why?

Practical 6
Error correction using nanopolish

Error correction using nanopolish

Nanopolish is another free open-source tool for nanopore data analysis. In contrast to tools such as racon it uses not only the sequence information from fasta/fastq files but also utilises the raw signal of each read that is stored in the fast5 files: it compares the raw signal of each nucleotide in a dataset and tries to identify incorrect base calls as well as DNA modifications such as methylation.

The recommended workflow to polish and existing draft assembly includes the following steps:

1. Nanopolish index	Use nanopolish to link the raw fast5 signal with base called fastq sequences
2. BWA index	Prepare your draft genome for use with read aligner BWA
3. Read mapping	Use BWA together with samtools to map the base called reads back to your draft genome
4. Nanopolish consensus	Build an error corrected consensus

For large assemblies (>50K) nanopolish provides two scripts *nanopolish_makerange.py* and *nanopolish_merge.py* to split the sequence into multiple segments before the error correction steps and merging them again afterwards (see http://nanopolish.readthedocs.io/en/latest/quickstart_consensus.html for more details).

Error correction using nanopolish

1. Nanopolish index

First change into the prac6 directory. To index the nanopore reads, i.e., to connect the fastq sequences with the corresponding raw signal first concatenate all fastq files in the albacore_pass directory into one fastq file using the linux command *cat*:

```
Cd ../prac6
cat ../prac1/albacore_out/workspace/pass/* > ./all_pass.fastq
```

This will write one fastq file into the prac6 folder containing all sequences that passed the initial albacore quality control.

Use nanopolish to index your fast5 and fastq reads

```
nanopolish index –d \
WORKSHOP_DIR/nanopore_practicals/precomp/MinKNOW/data/reads/20180419_0303/fast5/skip/0/ \
./all_pass.fastq
```

This command will create several files index files in the directory for practical 6.

2. BWA index

BWA is a widely used free open-source read mapper developed to quickly map sequences back to a reference genome. To speed up mapping of large read files *bwa* indexes the refence genome

Index the draft assembly using

```
bwa index ~/TOAST2018/day2/prac4/miniasm_filtered.fasta
```

Error correction using nanopolish

3. Map reads to draft assembly

As input for the mapped reads nanopolish expects a sorted and indexed *bam* file.

Use *bwa* to create an output file in bam format and the free open-source package *samtools* to subsequently sort and index it.

```
bwa mem –x ont2d ../prac4/miniasm_filtered.fasta ./all_pass.fastq \
| samtools sort -o ./reads.sorted.bam

samtools index reads.sorted.bam
```

4. Run nanopolish

Use the mapping information together with the linked fastq and fast5 files to create a polished assembly

```
nanopolish variants --consensus ./nano.fasta -r ./all_pass.fastq \
-b ./reads.sorted.bam -g ../prac4/miniasm_filtered.fasta
-q dcm,dam --min-candidate-frequency 0.1
```

Note

This nanopolish call will run for several hours. You can find the final file *nano.fasta* in directory WORKSHOP_DIR/*nanopore_practicals /precomp/data*.

Error correction using nanopolish

The above call will call nanopolish in *methylation aware* mode. DNA modifications can lead to errors in the base calling due to the fact that modified nucleotides show a different raw signal than their unmodified counterparts. By using option *-q dcm,dam* nanopolish is trying to determine the differences in the signal of 5-methyle-cytosine and unmodified cytosines and use this information to correct potential base calling errors.

As before, use dnadiff to assess the nanopolish results and compare them with racon:

```
dnadiff WORKSHOP_DIR/nanopore_practicals/precomp/data/chr17.fasta \
-p dd_con3 ../prac5/consensus_assembly2.fasta
```

What are the differences?

If you run racon on the polished data does it improve?

Note

Despite the systematic errors of nanopore base callers sequencing depth as well as organism can affect nanopolish results. The current eukaryotic test data set has a mean sequencing depth of <10. Nanopolish results of >99% accuracy have been reported for bacterial data set with >15 x coverage. However, to achieve best performance from nanopolish a sequencing depth of >=65 is recommended.

Practical 7
Canu assembly quality

Canu assembly quality

In contrast to miniasm the nanopore assembler *Canu* combines unitig assembler and contig assembly and also provides error correction. Recent assembler comparisons on bacterial data indicated higher accuracy and lower error rates for canu assemblies. However, canu's accuracy comes the cost of significant increases in run time and computational resources.

In directory *WORKSHOP_DIR/nanopore_practicals/precomp/data* you can find the canu assembled contigs of the tutorial data in file *canu.contigs.fasta.*

Additionally, the same directory includes the the largest if the canu assembled contigs polished with nanopolish in file *canu.contigs.polished.fasta*.

The commands used to assemble and polish the the sequences are:

```
Canu –p OUTPUT_PREFIX –d output_directory genomeSize=0.6m \
--nanopore-raw READS.FASTQ useGrid=false maxThreads=NO_CPUS \
maxMemory=RAM

bwa index CANU_ASSEMBLY

bwa mem –x ont2d –t THREADS READS.FASTQ |samtools sort –o \
OUTPUT_BAM
samtools index OUTPUT_BAM

nanopolish variants --consensus OUTPUT_FASTA –r READS.FASTQ \
-g CANU_ASSEMBLY –b READS.BAM –min-candidate-frequency 0.1\
-q dcm,dam
```

Hope you enjoyed these practical.

For questions and improvement suggestions please contact me at tim.kahlke@uts.edu.au