

TRAINING COURSE IN Computational Methods for Epitranscriptomics

Bari, 26th-28th April 2023



From raw data to RNA modifications: how the analysis works

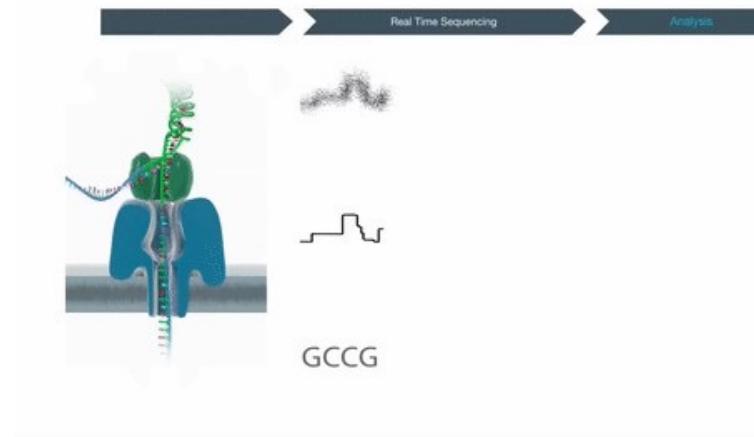
Introduction to the bioinformatics concepts, pipelines and software tools to analyse Nanopore data in order to detect RNA modifications.

Camilla Ugolini

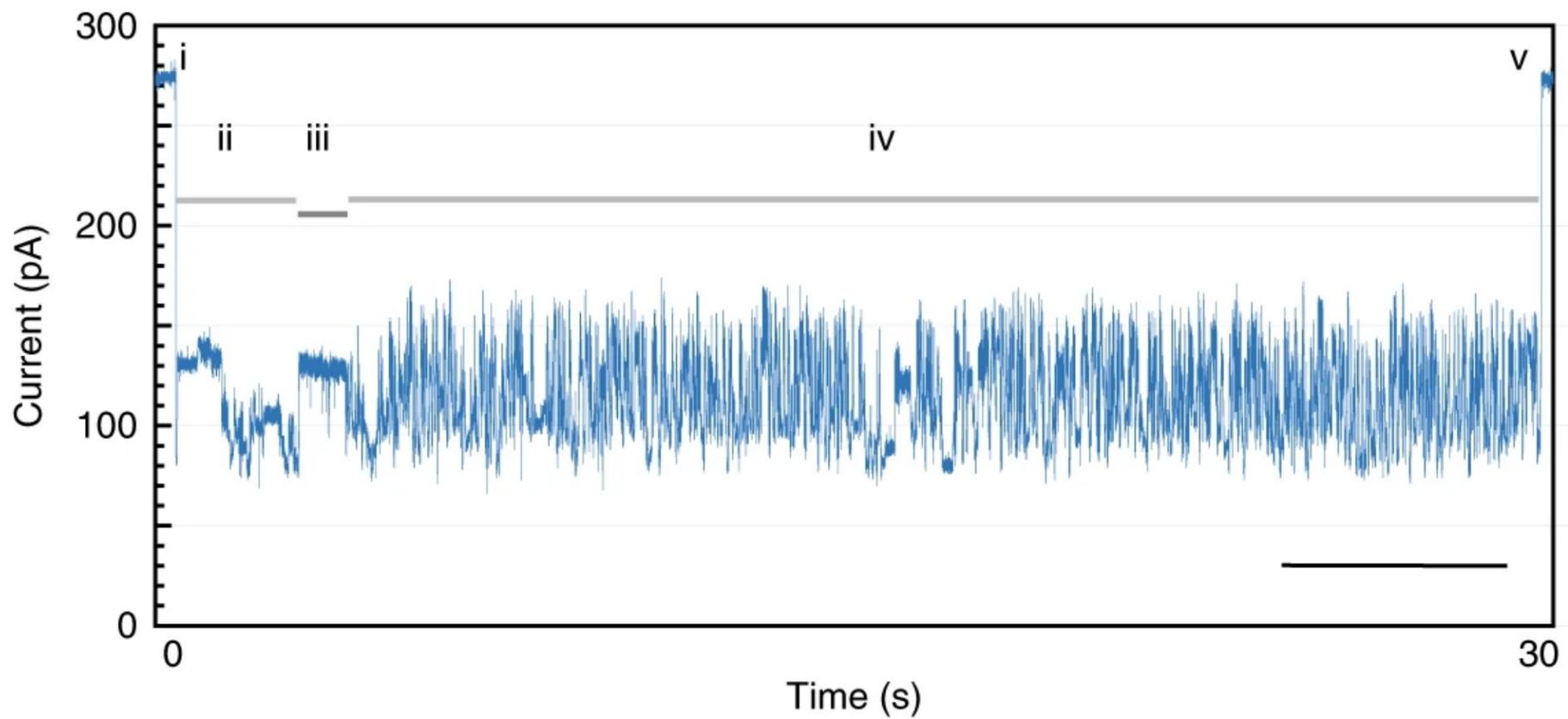
Credits: Tommaso Leonardi & Luca Pandolfini

Nanopore data

- Raw data is current (pA) over time
- RNA moves through the pore at 70bps
- The current is recorded at 4kHz
- ~57 data points per kmer
- Data stored in Fast5 format (HDF5)



The squiggle

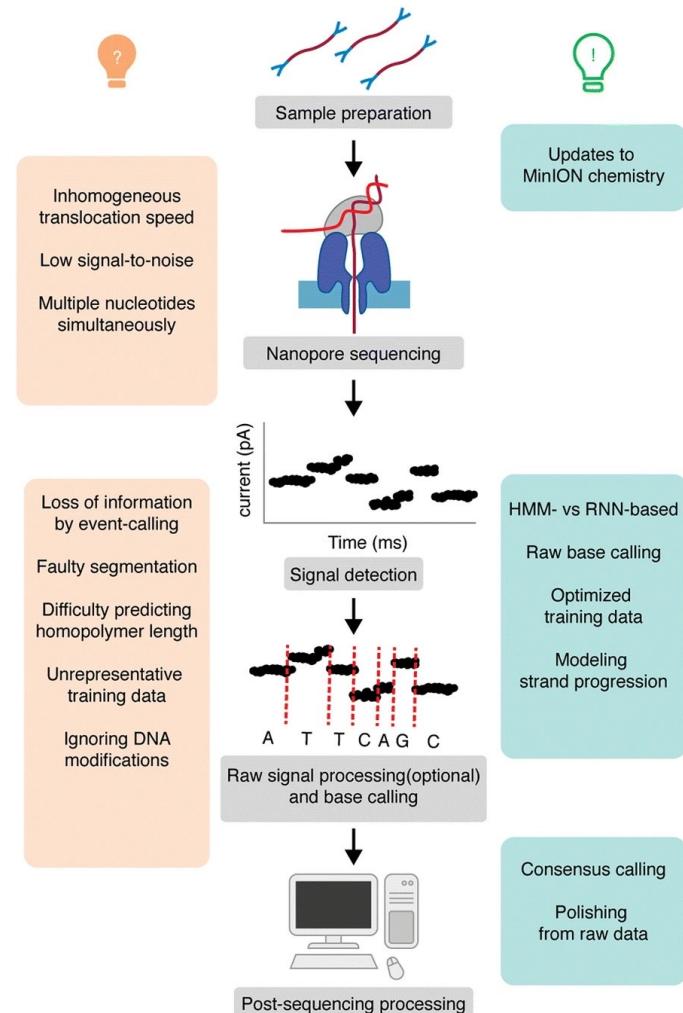


Workman et al., Nat Meth 2019

From the sample to the data

Multiple factors influence the properties and quality of the data:

- Library chemistry
- Electronics
- Pore
- Basecalling



Analysis steps

<input checked="" type="checkbox"/> Sequencing	Starting and monitoring a run
<input type="checkbox"/> Basecalling	From signal to sequence
<input type="checkbox"/> QC	Assess quality of the data
<input type="checkbox"/> Mapping	Map the reads to the genome/transcriptome
<input type="checkbox"/> Quantification	Estimate gene/transcript expression
<input type="checkbox"/> Modification detection	Detect RNA modifications

Current basecalling algorithm

- Guppy is the current production basecaller
- It's included in MinKnow
- Based on RNN
- Trained on a collection of human, yeast, *E. coli* and *C. elegans* samples
- Implements a fast model and an Higher Accuracy model (HAC)
- Supports GPU acceleration



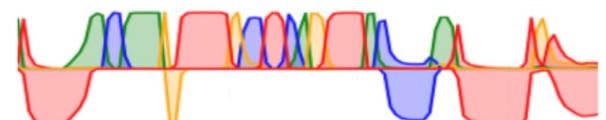
Raw signal



Recurrent Neural Network
(over timesteps)



Base to base
transitions
(per time-step)



T+ T- A+ C+ A+ G+ T+ G+ C+ T+ C+ A+ G+ T+ A+ C+ A+ T+ C- T- T+ G+ T+ T-

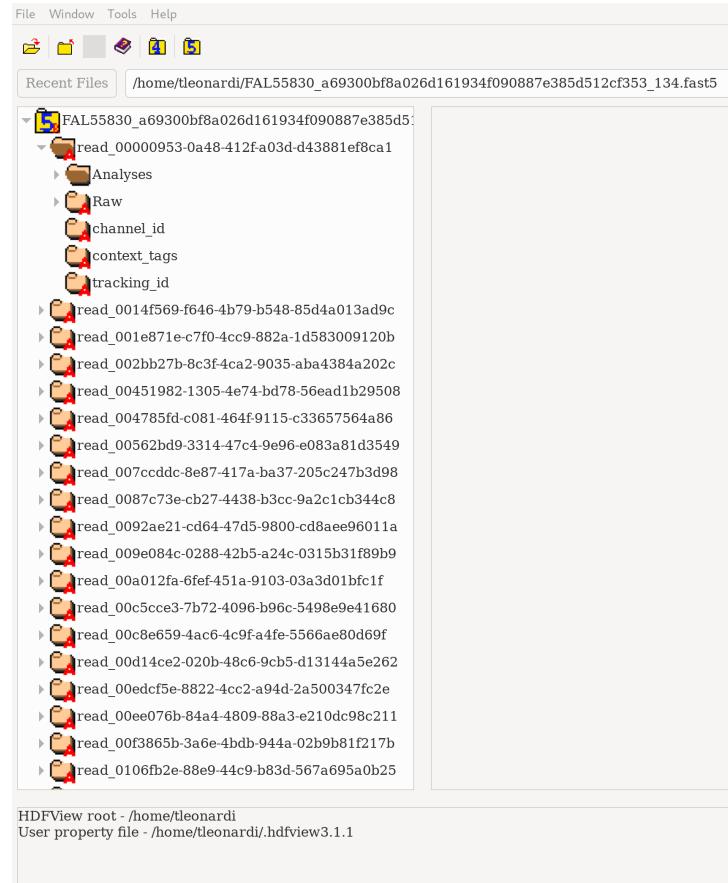
TTACAGGTGCTCAGTACCAATTGTT

Per-flip-flop
Base probabilities

Basecall

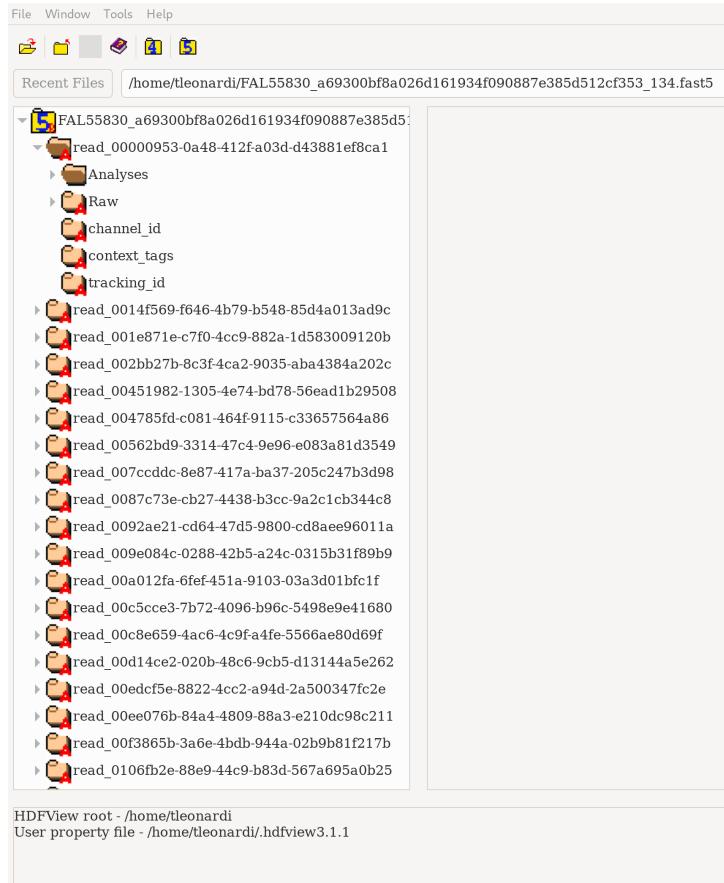
Fast5 files

- Binary format based on HDF5
- Slots contain various types of data/metadata
- Always contain raw signal
- Can also contain basecalled sequence
- One file can contain data for multiple reads (multi fast5)
- Default: 1 fast5 -> 4000 reads



Working with fast5 files: hdfview

- Download one fast5 produced by the Nanopore WGS consortium:
<https://github.com/nanopore-wgs-consortium/NA12878/blob/master/NA12878.fast5>
- Load hdfview and open the file



Basecalling with Guppy

Guppy is free to use but it is not open source. It can be downloaded from the ONT Community website after registering.

Important Options

Flag	Function
-i input path	Path to the root folder containing fast5 files to basecall
-s output path	Path to the output folder
--flowcell name	Flowcell model used (e.g. FLO-MIN106)
--kit name	Kit used for library prep (e.g. SQK-RNA002)

Guppy: other important options

Important Options

Flag	Function
-- qscore_filtering	Flag that enables pass/fail filtering
--min_qscore int	Min q-score to consider a read as pass
--calib_detect	Filter calibration strand reads in a dedicated folder
-- reverse_sequence	Reverse the called sequence (used for RNA sequencing due to 3' to 5' direction)
--u_substitution	Substitute 'U' for 'T' in the called sequence
--recursive, -r	Look for fast5 files recursively inside the input path

Guppy: less important options

Important Options

Flag	Function
--trim_strategy	Strategy to trim adapters: dna, rna or none
--disable_pings	Do not send telemetry info to ONT
--compress_fastq	Compress fastq files with gzip
--fast5_out	Save basecalled fast5 files
--resume	Resume a stopped run

Guppy input/output

Input: directory containing fast5 files to basecall

Output:

Guppy output location (MinKnow)

Guppy stores its output at:

Var	Meaning
output_dir	base output directory
experiment_id	experiment id specified when starting the run
sample_id	sample_id specified when starting the run
start_time	time when the run started (YYYYMMDD_HHMM)
device_id	serial ID of the MinION or device position for GridION/PromethION (e.g. X1-5)
flow_cell_id	flow cell ID

Guppy summary file

A tsv file containing useful information on the run

Guppy: basecalling on CPU

Important Options

Flag	Function
--num_callers	Number of parallel basecaller processes
--num_cpu_threads_per_caller	Number of CPU worker threads per basecaller

- The total number of threads used is `num_callers * cpu_threads_per_caller`
- The total number of threads should not exceed the number of cores at your disposal
- Guppy uses 4GB of RAM plus 1 extra GB per thread

Guppy: basecalling on GPU

GPU basecalling gives a significant speed up. Use it whenever possible.

Important Options

Flag	Function
--device cuda:0	Instructs Guppy to use the first GPU
--chunks_per_caller	Number of signal chunks to collect before sending to basecaller

Supported GPUs:

- NVIDIA Tesla V100
- NVIDIA Quadro GV100
- NVIDIA Jetson TX2
- NVIDIA Jetson Xavier
- Other GPUs w/ CUDA Compute Capability >6.1 (limited support)

Other tools and resources

ONT provides various open source tools for analyses and manipulation of sequencing data and they are available on github.com/nanoporetech/. The following is a non exhaustive list of some interesting ones:

- Taiyaki: model training ([nanoporetech/taiyaki](https://github.com/nanoporetech/taiyaki))
- Bonito: research open source basecaller ([nanoporetech/bonito](https://github.com/nanoporetech/bonito))
- Dorado: latest beta production basecaller ([nanoporetech/dorado](https://github.com/nanoporetech/dorado))
- Rerio: experimental basecalling models for Guppy ([nanoporetech/rerio](https://github.com/nanoporetech/rorio))
- Pychopper: orient and trim cDNA reads ([epi2me-labs/pychopper](https://github.com/epi2me-labs/pychopper))
- Tombo: modified base detection with dRNA support ([nanoporetech/tombo](https://github.com/nanoporetech/tombo))

Analysis steps

<input checked="" type="checkbox"/> Sequencing	Starting and monitoring a run
<input checked="" type="checkbox"/> Basecalling	From signal to sequence
<input type="checkbox"/> QC	Assess quality of the data
<input type="checkbox"/> Mapping	Map the reads to the genome/transcriptome
<input type="checkbox"/> Quantification	Estimate gene/transcript expression
<input type="checkbox"/> Modification detection	Detect RNA modifications

Data QC

Quality controls give an immediate feedback on the performance of the run through a number of metrics:

- Experiment throughput
- Read length
- Read quality
- Error rate
- Flowcell stats (yield over time, n. pores, etc)



Where is the QC info stored?

Metrics for assessing QC can be retrieved from multiple locations:

- Sequencing summary file
- Fast5 files
- BAM files (after mapping)
- Telemetry files

QC tools

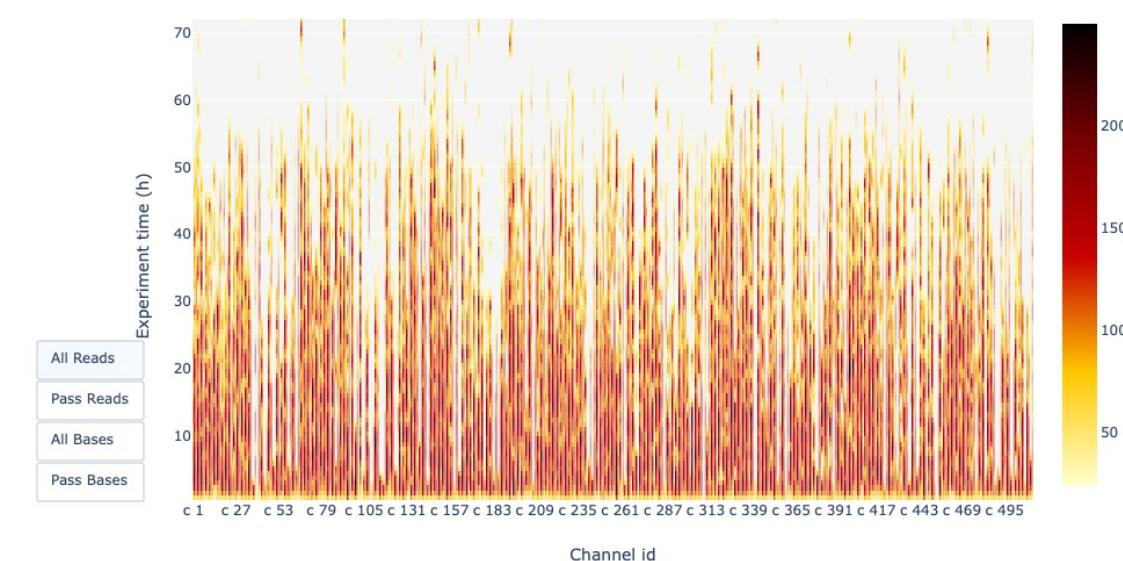
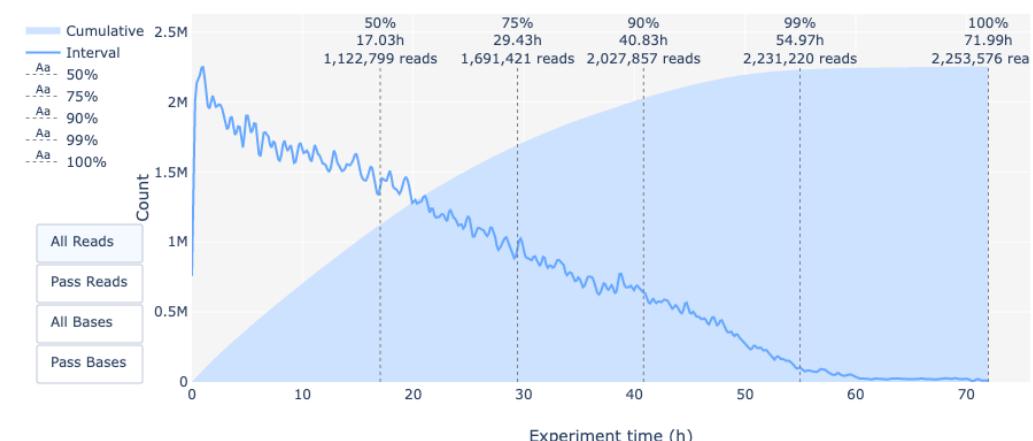
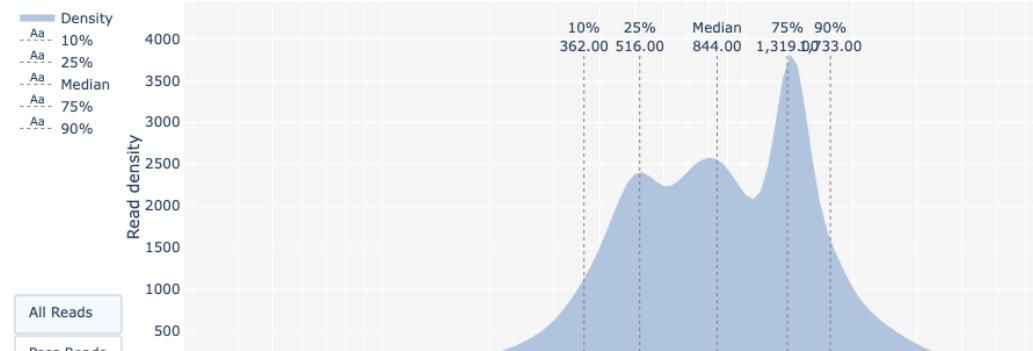
Several tools take care of collecting and plotting QC data:

Pipeline	Github Repo
★ pycoQC	a-slide/pycoQC
★ nanoQC	wdecoster/nanoQC
☆ LongQC	yfukasawa/LongQC
☆ minion_qc	roblanf/minion_qc



Features:

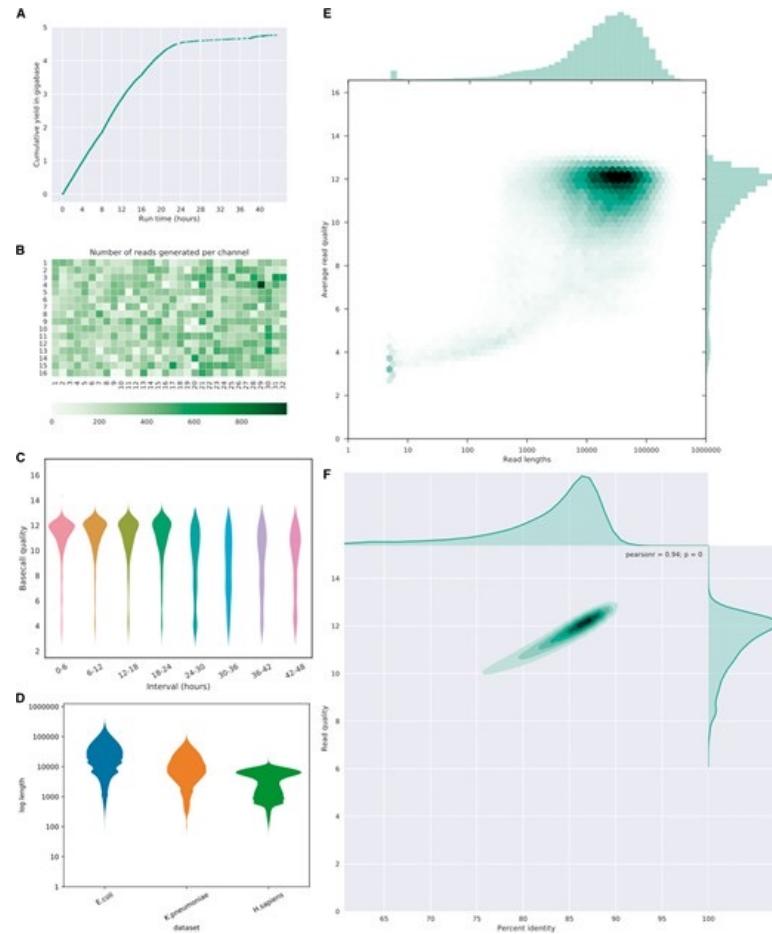
- Reads sequencing_summary or fast5s
- Very fast
- html report with interactive plots
- Optionally supports reading BAM file for alignment/identity stats



nanoQC

Features:

- Reads sequencing_summary, fast5s, fastq or BAMs
- Very fast
- Generates bivariate plots of read lengths, quality and ref identity



De Coster et al., Bioinformatics 2018. doi: 10.1093/bioinformatics/bty149

Running pycoQC

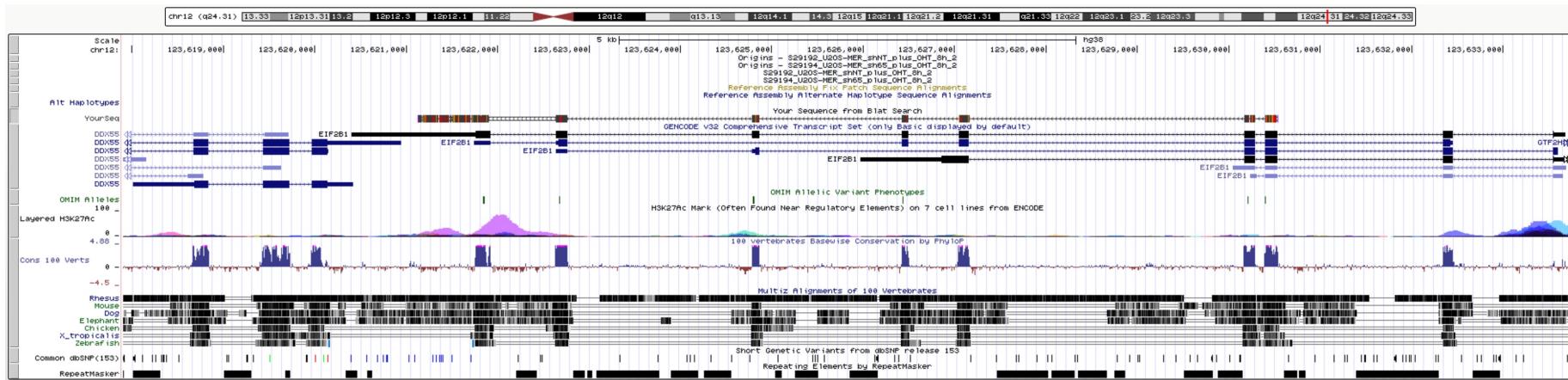
Main Options

Flag	Function
--summary_file sequencing_summary.txt	Path to summary file generated by Guppy
--html_outfile output path	Path to the output html file
--filter_calibration	Remove reads marked as calibration by Guppy

Analysis steps

<input checked="" type="checkbox"/> Sequencing	Starting and monitoring a run
<input checked="" type="checkbox"/> Basecalling	From signal to sequence
<input checked="" type="checkbox"/> QC	Assess quality of the data
<input type="checkbox"/> Mapping	Map the reads to the genome/transcriptome
<input type="checkbox"/> Quantification	Estimate gene/transcript expression
<input type="checkbox"/> Modification detection	Detect RNA modifications

Mapping the reads



- Nanopore error rate is higher than Illumina
- Reads are very long (up to Mb)
- Need optimised mappers
- Most common mapper: Minimap2
- Reads can be mapped to genome and/or transcriptome
- Output: BAM file

Genome vs transcriptome

Genome

Advantages

- Detect new genes/transcripts
- Visualise data on a genome browser

Disadvantages

- Relies on good annotation
- Requires splice-aware alignment
- Assigning reads to individual TXs is complex

Transcriptome

Advantages

- Direct assignment of read to transcripts
- Trivial quantification of isoforms
- No need to consider splicing

Disadvantages

- Limited to annotated genes/transcripts
- Can't be loaded on a genome browser

Assemble the Transcriptome

Long reads are mostly full-length: assembly step is *in theory* not required
But they suffer from higher error rates, that makes building a consensus more difficult

Scenario:

- A *genome* is not available or it is not desirable to take it into account
- A *transcriptome* is not available or it is not desirable to take it into account

Final goals:

- To cluster long read RNAseq reads into groups corresponding to *gene families*
- To obtain a collection of *consensus transcripts*

Assemble the Transcriptome

Tool	Github Repo
★ stringtie2	skovaka/stringtie2
★ Flair	BrooksLabUCSC/flair
★ TALON	mortazavilab/TALON
★ Pinfish	nanoportech/pinfish

Evaluate Transcriptome Quality

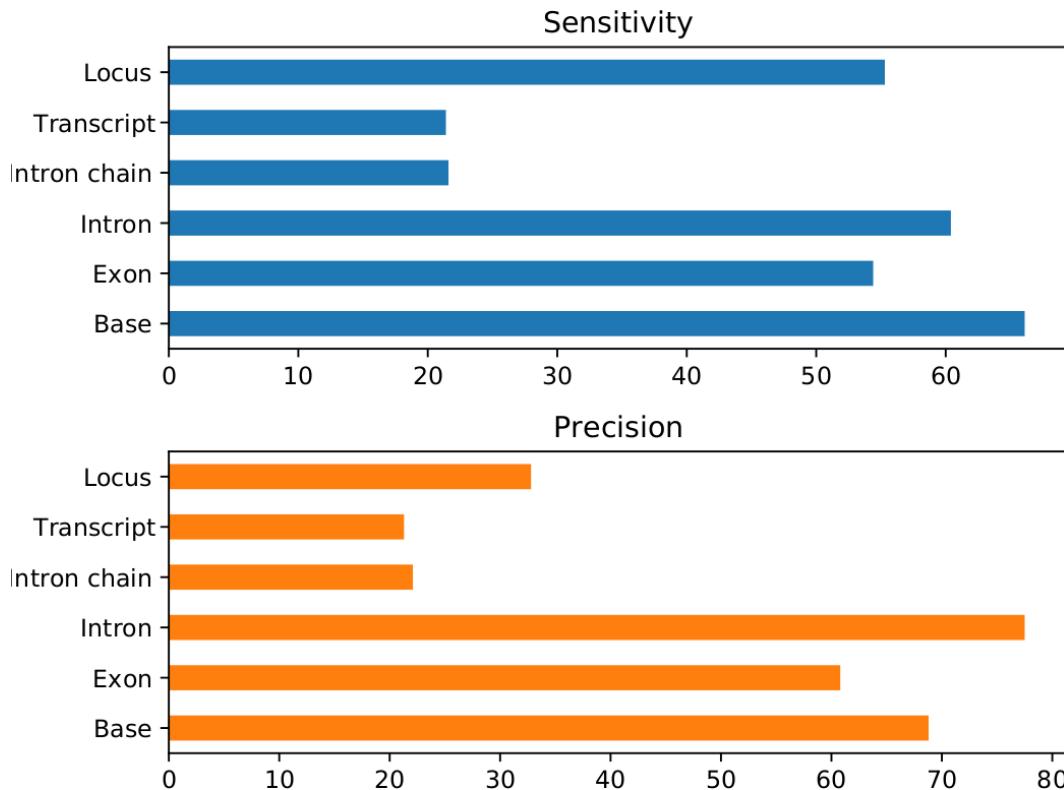
Gffcompare

Estimate accuracy of one or more GFF files (the “query” files) when compared with a reference annotation (also provided as GFF)

(Pertea et al., 2020; doi:10.12688/f1000research.23297.1)

Evaluate Transcriptome Quality

Sensitivity and Precision of the Assembly



Per-item statistics:

$$\text{Sensitivity} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

- **TP (true positives)**: total bases that are reported on both the query and **any** reference item
- **FN (false negatives)**: total bases in reference by any of the predicted items
- **FP (false positives)**: total bases covered by an but not covered by any reference item

Input data: the fastq format

Read name	Sequence	Quality
@32e69ca0-0744-4020-8956-17ab085a841e runid=2749b read=100 ch=207 start_time=2017-10-10T06:56:00Z	GGAGAGGGAGGAAAGCACATTGGGAACGAGTTCTCAAACTAATGCTCAGAACCGGTGTCCAGTGACGTAGAAACCACTTCCAAATGGCGGGCTTTTC	
+		
97; ;9:999988; : ; ;8=;86;89;97::978799; ;778:9:9688:9:9;4:579784785671967264322538626226/.2350412310100;		

Phred scores

@32e69ca0-0744-4020-8956-17ab085a841e runid=2749b read=100 ch=207 start_time=2017-10-10T06:56:00Z
 GGAGAGGGAGGAAAGCACATTGGAACGAGTTCTAAACTAATGCTCAGAACCGGTGTCCAGTAGAAACCACTTCCAAATGGCGGGCTTTC
 +
 97;;9:999988;:::;8=;86;89;97::978799;;778:9:9688:9:9;4:579784785671967264322538626226/.2350412310100;

ASCII_BASE=33 Illumina, Ion Torrent, PacBio and Sanger											
Q	P_error	ASCII	Q	P_error	ASCII	Q	P_error	ASCII	Q	P_error	ASCII
0	1.00000	33 !	11	0.07943	44 ,	22	0.00631	55 7	33	0.00050	66 B
1	0.79433	34 "	12	0.06310	45 -	23	0.00501	56 8	34	0.00040	67 C
2	0.63096	35 #	13	0.05012	46 .	24	0.00398	57 9	35	0.00032	68 D
3	0.50119	36 \$	14	0.03981	47 /	25	0.00316	58 :	36	0.00025	69 E
4	0.39811	37 %	15	0.03162	48 0	26	0.00251	59 ;	37	0.00020	70 F
5	0.31623	38 &	16	0.02512	49 1	27	0.00200	60 <	38	0.00016	71 G
6	0.25119	39 '	17	0.01995	50 2	28	0.00158	61 =	39	0.00013	72 H
7	0.19953	40 (18	0.01585	51 3	29	0.00126	62 >	40	0.00010	73 I
8	0.15849	41)	19	0.01259	52 4	30	0.00100	63 ?	41	0.00008	74 J
9	0.12589	42 *	20	0.01000	53 5	31	0.00079	64 @	42	0.00006	75 K
10	0.10000	43 +	21	0.00794	54 6	32	0.00063	65 A			

ASCII_BASE=64 Old Illumina											
Q	P_error	ASCII	Q	P_error	ASCII	Q	P_error	ASCII	Q	P_error	ASCII
0	1.00000	64 @	11	0.07943	75 K	22	0.00631	86 V	33	0.00050	97 a
1	0.79433	65 A	12	0.06310	76 L	23	0.00501	87 W	34	0.00040	98 b
2	0.63096	66 B	13	0.05012	77 M	24	0.00398	88 X	35	0.00032	99 c
3	0.50119	67 C	14	0.03981	78 N	25	0.00316	89 Y	36	0.00025	100 d
4	0.39811	68 D	15	0.03162	79 O	26	0.00251	90 Z	37	0.00020	101 e
5	0.31623	69 E	16	0.02512	80 P	27	0.00200	91 [38	0.00016	102 f
6	0.25119	70 F	17	0.01995	81 Q	28	0.00158	92 \	39	0.00013	103 g
7	0.19953	71 G	18	0.01585	82 R	29	0.00126	93]	40	0.00010	104 h
8	0.15849	72 H	19	0.01259	83 S	30	0.00100	94 ^	41	0.00008	105 i
9	0.12589	73 I	20	0.01000	84 T	31	0.00079	95 _	42	0.00006	106 j
10	0.10000	74 J	21	0.00794	85 U	32	0.00063	96 ~			

$$Q = -10 \log_{10} P$$

$$P = 10^{\frac{-Q}{10}}$$

Phred score	Probability that the base is incorrect	Precision of the base
10	1 in 10	90 %
20	1 in 100	99 %
30	1 in 1000	99.9 %
40	1 in 10000	99.99 %

Real reads

```
@32e69ca0-0744-4020-8956-17ab085a841e    runid=2749b    read=100    ch=207    start_time=2017-10-10T06:56:00Z
UGGCAAGCACCUGGGCUCCGCAGCAGGAAGCCUCCGGCCCAGUAUAGGAAAUACUUUUGAAGUUCAGAUGAAAGUGGAUUCUGAUGCACGGCCU
GCAAGUGGCCGCCACUCCGGACGUGCUGGAGUACUGAAGAAUAAAAGGGGAGUUGCACAGGGUCUGGGCGAACUUACCAGUGCCGCAGAAACCUGU
GUGGUGGUGUGACUCCUUCUGUGAAGUGUCCUCUGGCGGGGAGCUUUCUCCGCUUCAUCAGUUUUGCCUCCUGGAAAUACUCCGAUUACUCCAA
AUGUAAAAAGAACAUUGAGCGGGGGGGGAGAACUUUUUCAGGAUACUGUCAAGAACAAAUAAGAUCUGUGCCAUAUUCAUCAAAG
AUGGAGCGACAAAUAUUGACUUUCACGCCUACUCCAGAGUGACUGAGAGUCUGGAAGCAGCCGUGGCCAGAACAGCGAUUAGUGUAUACGUUAC
AGAGUCACAGCCUGAUUUUGUCAGGUAGAAAAUGGCCAAGCCUUUUGCCACCUUCAACGUCCUGUCACUGUGGUGCUAGAUGCUGCUGUCGGUA
CAUCAUGGAGAAAGCAGAUUUUGUUUAGUUGGUGCUGAAAGGAGUUGUUGAAACGGAGGAAUUAACAAGAUUGGAACCAACCAGAUGGCUGUGU
GUGCCAAGCACAGAACAAACCUUCAUGUGGUUGCAGAAAGUUCUGGUUUUGUCCGGCUCUCCACUAAACCAGCAAGACGUCCAGAUAAAGUUA
GUUAACGCUGAUAGACCAUUUCCAAGGUCGCGCAGACUGGACAAGACCUUCAAAGAGGGAGCAUCCGUGGCCACUAGUCCCCUCCUUAUCACUC
UCUGCUGUUAUAGACCUGGGUGCUGACACCACAGCAGCAGCGAUGAGCUCUAUCAAGCUCUACUGUACCUUGAGGCCUUUCCUGCCAAGGUGC
AGCUUUACGUAGUUAGGCAGGUGAGUAGCUGCUUGACACCCAGUGAAUAGGCCAAACUGAGAUGGUUUUAUGAAGAUCUAUGGAGUAAGGCUUA
AAAUCAUCAUUUGGAGAUUCUACUACGUCCAUCUAAAUGUGUUCAGCUAUUUAAAUCCCAACUUAAAUCGGUUUCUAGAAAACUUUC
CUUUUUCAGUUUCACCAGAGCUACAAGUUAGAUUUUGUUAUGAAAGAUGAGGCCUAGGUCCACCUUCCUGCACUCCGG
CUGAUCUGCCAAAGCACGCAAGAUGCAGGCAGCAGCCAUACCCUCUGCCACAAACGACCAGCUGGUACAGCGGGUGCCAUGUAA
GAGGCAAGAACACUUGCCGAAUCUGCGUCUGGUACAGUGAAGCACAUCCAGCAGGAUCAGCCAAACAGUAAAACUACCAAGAGAACGAGGAA
AGCAGAAAAAACGAUGUUUAGCAACACAGUAUUCUGCAUGGUUCUUGUUUAAGAAAAUGCCUUCUGAAUAUUUGUAACUGAAAUCUGUAUGUGU
UAAAC
+
#$(($'-+*(,6+. ,3#${(1,0///0-%**.(4#%*)*03,"'%%#.0./:1-,(/@..(+0*,-.((,.1.0(%$213().**%#"%"($%('5/%
%$.%)%2(6+-0%)(54:4,+(+,/83=)*."$))17*+/(015-'2*')$##$/%)??++1,41+,.:(%'*.-+,.22;/"#)*/?4.40+*
....
```

Genome mapping: minimap2

- Genome fasta can be downloaded from Ensembl
- You can find it in ~/data

Important Options

Flag	Function
-a	Output alignments in SAM format
-x splice	Use a preset of options that allow spliced alignments
-k14	Use a smaller k-mer size, recommended for noisy dRNA-Seq reads
-uf	Consider forward strand only when looking for canonical splice-sites (GT-AG)

Genome mapping: inspect results

Results are stored in SAM format
Samtools can be used to view/filter the file

Important Options

Field	Definition
1	QUERY name
2	FLAG
3	REFERENCE position
4	Mapping position
5	Mapping quality
6	CIGAR string
..	Additional mandatory fields up to 11

Transcriptome mapping step 1: generate the transcriptome

- Genome fasta and transcriptome GTF can be downloaded from Ensembl
- You can find them in ~/data

Transcriptome mapping step 2: minimap2

Important Options

Flag	Function
-a	Output alignments in SAM format
-x map-ont	Use a preset combination of options optimised for ONT data
-k14	Use a smaller k-mer size, recommended for noisy dRNA-Seq reads
--for-only	Only map to the forward strand of the reference

SAM/BAM filtering

The SAM files generated by minimap2 usually need to be filtered for:

- Unmapped reads
- Secondary alignments
- Supplementary alignments

Filtering can be done using the SAM FLAG

Analysis steps

<input checked="" type="checkbox"/> Sequencing	Starting and monitoring a run
<input checked="" type="checkbox"/> Basecalling	From signal to sequence
<input checked="" type="checkbox"/> QC	Assess quality of the data
<input checked="" type="checkbox"/> Mapping	Map the reads to the genome/transcriptome
<input type="checkbox"/> Quantification	Estimate gene/transcript expression
<input type="checkbox"/> Modification detection	Detect RNA modifications

Expression quantification: a naïve approach

You can just count how many reads map to each gene!

Expression quantification: Nanocount

Our naïve above completely ignores the problem of multimapping reads (we filtered out all secondary alignments). Dedicated tools such as **Nanocount** take secondary alignments into account using Expectation Maximization algorithms, providing more robust estimates of transcript abundance.

transcript_name raw	est_count	tpm	
ENST00000362079(+) 0.028630405163205842	30111.99999999658	28630.405163205844	
ENST00000361739(+) 0.02047731177272376	21536.992179650442	20477.311772723762	
ENST00000361624(+) 0.02046778636250834	21526.973838981787	20467.78636250834	
ENST00000387347(+) 0.0143237597563661	15064.99999999829	14323.7597563661	
ENST00000343262(-) 0.014215488295511594	14951.125599316025	14215.488295511594	
ENST00000361899(+) 0.013645841355683124	14351.99999999837	13645.841355683124	
ENST00000361381(+) 0.011135736758483887	11711.99999999867	11135.736758483887	
ENST00000361390(+) 0.010168776010244636	10694.999999998785	10168.776010244635	
ENST00000233143(+) 0.010080351871025118	10601.999999998796	10080.351871025117	

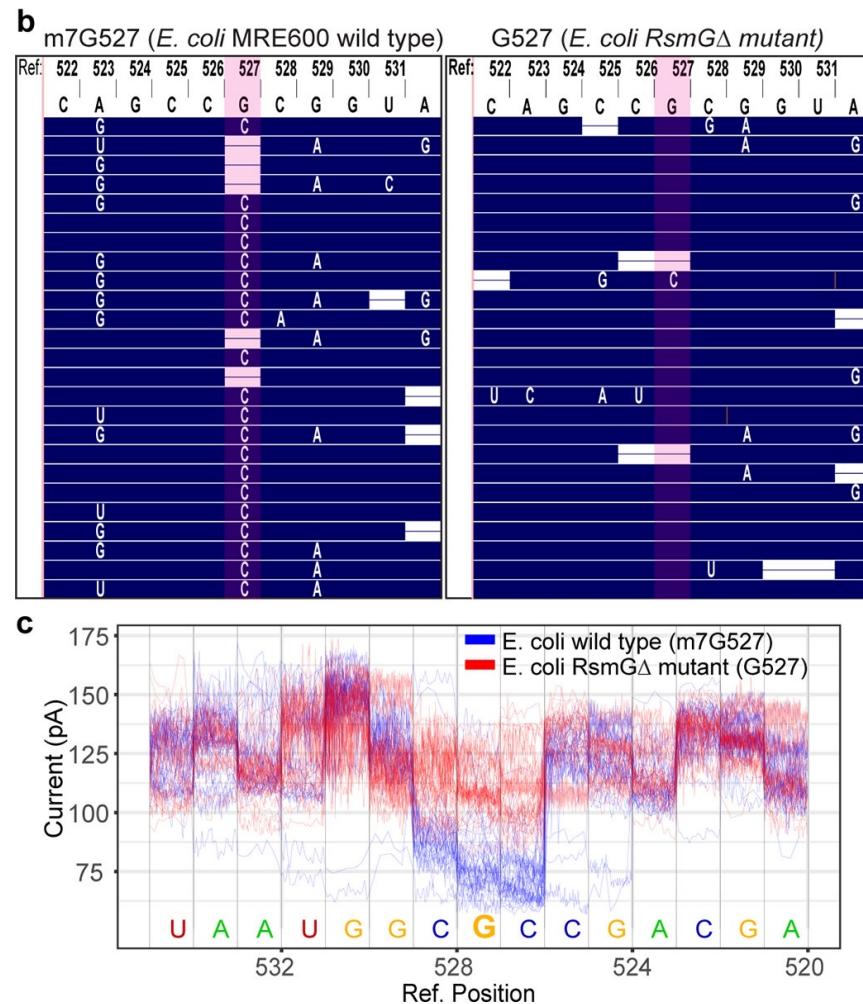
Analysis steps

<input checked="" type="checkbox"/> Sequencing	Starting and monitoring a run
<input checked="" type="checkbox"/> Basecalling	From signal to sequence
<input checked="" type="checkbox"/> QC	Assess quality of the data
<input checked="" type="checkbox"/> Mapping	Map the reads to the genome/transcriptome
<input checked="" type="checkbox"/> Quantification	Estimate gene/transcript expression
<input type="checkbox"/> Modification detection	Detect RNA modifications

Connecting the signal to the sequence

Why bother?

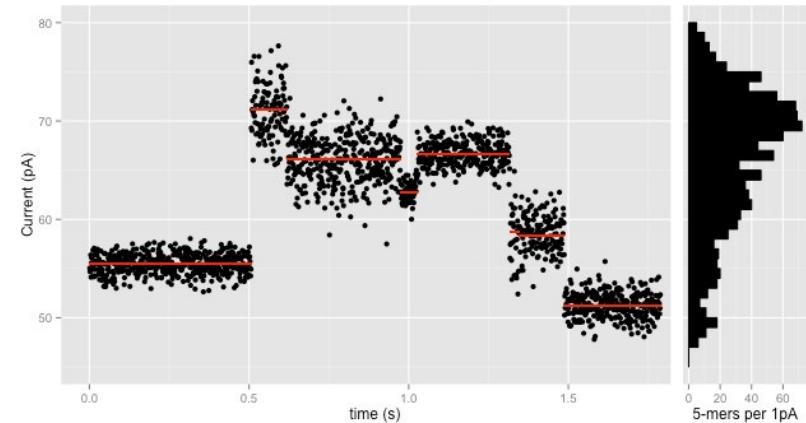
- Visualisation
- Identification of modifications
- Detection of basecalling inaccuracies
- Comparison of samples



Connecting the signal to the sequence

Why is it hard?

- Would be (was) easier with older basecalling algorithms
- Current NN don't provide a direct "association" between signal and seq
- Resquiggling needs to be done a posteriori using HMM
- Tool of choice: **Nanopolish**



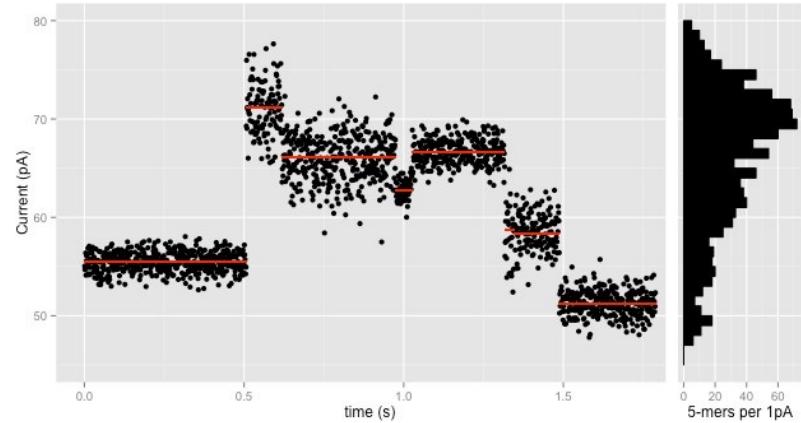
simpsonlab.github.io

Nanopolish eventalign

Align signal-level events to k-mers of a reference genome

Input

- Basecalled sequences
- Reference sequence (i.e. mapped reads)
- Raw data (fast5 files)



simpsonlab.github.io

Nanopolish eventalign

```
> nanopolyh eventalign
Usage: nanopolyh eventalign [OPTIONS] --reads reads.fa --bam alignments.bam --genome genome.fa
Align nanopore events to reference k-mers
  -v, --verbose                                display verbose output
  --version                                     display version
  --help                                         display this help and exit
  --sam                                         write output in SAM format
  -r, --reads=FILE                             the ONT reads are in fasta FILE
  -b, --bam=FILE                               the reads aligned to the genome assembly are in bam FILE
  -g, --genome=FILE                            the genome we are computing a consensus for is in FILE
  -t, --threads=NUM                           use NUM threads (default: 1)
  --scale-events                             scale events to the model, rather than vice-versa
  -n, --print-read-names                     print read names instead of indexes
  --samples                                    write the raw samples for the event to the tsv output
  --signal-index                              write the raw signal start and end index values for the
                                             event to the tsv output
```

[Some options not relevant to this course were omitted to save space on the slide]

Main Options

Flag	Function
--scale-events	scale events to the model, rather than vice-versa
--print-read-names	print read names instead of indexes
--samples	write the raw samples for the event to the tsv output
--signal-index	write the raw signal start and end coordinates

Hands on: Building an index for Nanopolish

Nanopolish requires an index to associate reads to the corresponding fast5 files

1. Combine all the basecalled fastq files (~/*data/consortium_basecalled/pass/*) into a single one.
2. Run `nanopolish index`

```
# Combine all basecalled fastq files
> cat ~/data/consortium_basecalled/pass/*fastq > basecalled.fastq

> nanopolish index \
-d ~/data/consortium_basecalled/workspace/ \
-s ~/data/consortium_basecalled/sequencing_summary.txt \
basecalled.fastq

[readdb] indexing /home/ubuntu/data/consortium_basecalled/workspace/
[readdb] indexing /home/ubuntu/data/consortium_basecalled/workspace//1
[readdb] indexing /home/ubuntu/data/consortium_basecalled/workspace//2
[readdb] indexing /home/ubuntu/data/consortium_basecalled/workspace//4
[readdb] indexing /home/ubuntu/data/consortium_basecalled/workspace//3
[readdb] indexing /home/ubuntu/data/consortium_basecalled/workspace//0
[readdb] num reads: 11699, num reads with path to fast5: 11699

> ls
basecalled.fastq  basecalled.fastq.index  basecalled.fastq.index.fai
basecalled.fastq.index.gzi  basecalled.fastq.index.readdb
```

Hands on: Running eventalign

1. Convert the SAM file to BAM and sort it
2. Generate and index for the BAM file
3. Run eventalign with `--scale-events --print-read-names --samples --signal-index`

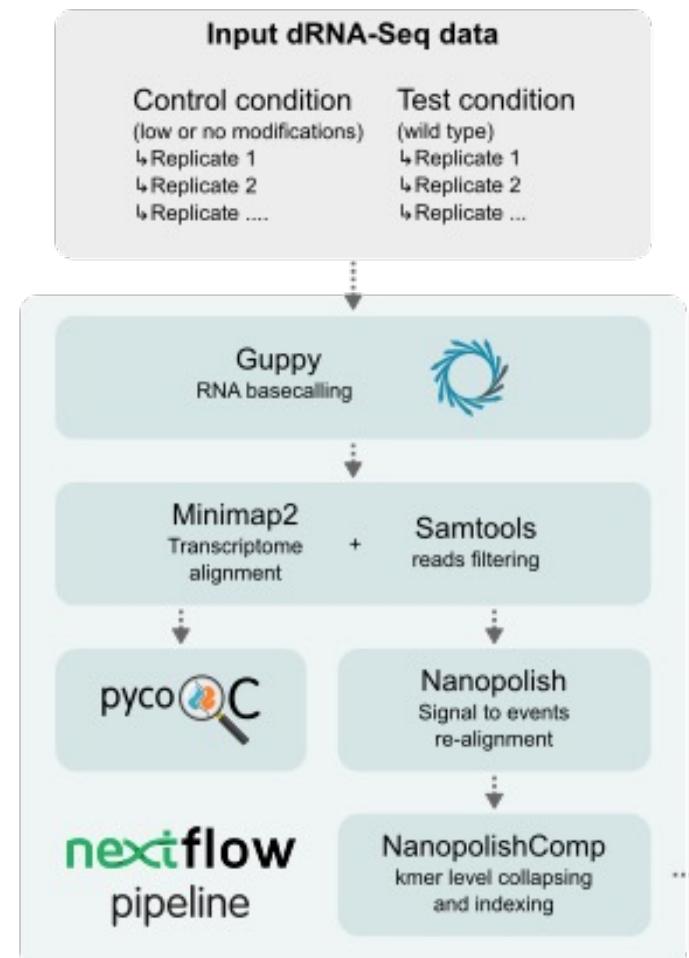
```
# Convert the SAM file to BAM and sort it
> samtools view \
  -b ~/data/consortium_minimap_transcriptome_filtered_oneread.sam | samtools sort > mapped_reads.bam

# Index the BAM file
> samtools index mapped_reads.bam

# Run nanopolish
> nanopolish eventalign \
  --scale-events --print-read-names --samples --signal-index \
  --reads basecalled.fastq \
  --bam mapped_reads.bam \
  --genome ~/data/Homo_sapiens.GRCh38.98_transcriptome.fa > eventalign.txt
```

How to go from the raw data to modification calls

- Basecall the fast5 files
- Map to the transcriptome
- Filter alignments
- Resquiggle with Nanopolish
- Collapse and index results with eventalign_collapse
- Analyse with Nanocompare



Leger et al., bioRxiv 2019

Step 1 (recap): basecalling with Guppy

```
> guppy_basecaller -i input_folder \  
    -s out_folder \  
    --flowcell FLO-MIN106 \  
    --kit SQK-RNA002 \  
    --fast5_out \  
    --recursive \  
    --num_callers 2 \  
    --disable_pings \  
    --qscore_filtering
```

Step 1 (recap): basecalling with Guppy

```
> guppy_basecaller -i input_folder \  
    -s out_folder \  
    --flowcell FLO-MIN106 \  
    --kit SQK-RNA002 \  
    --fast5_out \  
    --recursive \  
    --num_callers 2 \  
    --disable_pings \  
    --qscore_filtering
```

Expected output:

Step 1 (recap): basecalling with Guppy

```
> guppy_basecaller -i input_folder \  
    -s out_folder \  
    --flowcell FLO-MIN106 \  
    --kit SQK-RNA002 \  
    --fast5_out \  
    --recursive \  
    --num_callers 2 \  
    --disable_pings \  
    --qscore_filtering
```

Expected output:

1. fast5 files (basecalled)

Step 1 (recap): basecalling with Guppy

```
> guppy_basecaller -i input_folder \  
    -s out_folder \  
    --flowcell FLO-MIN106 \  
    --kit SQK-RNA002 \  
    --fast5_out \  
    --recursive \  
    --num_callers 2 \  
    --disable_pings \  
    --qscore_filtering
```

Expected output:

1. fast5 files (basecalled)
2. fastq files

Step 1 (recap): basecalling with Guppy

```
> guppy_basecaller -i input_folder \  
    -s out_folder \  
    --flowcell FLO-MIN106 \  
    --kit SQK-RNA002 \  
    --fast5_out \  
    --recursive \  
    --num_callers 2 \  
    --disable_pings \  
    --qscore_filtering
```

Expected output:

1. fast5 files (basecalled)
2. fastq files
3. sequencing_summary.txt

Step 2 (recap): Map to the transcriptome

Required Input: (1) fastq files, (2) transcriptome in fasta format

Step 2 (recap): Map to the transcriptome

Required Input: (1) fastq files, (2) transcriptome in fasta format

```
# Convert the Ensembl GTF to BED
bedparse gtf2bed \
~/data/Homo_sapiens.GRCh38.102.chr.gtf \
> reference_transcriptome.bed

# Generate a Fasta from the GTF
bedtools getfasta \
-fi ~/data/Homo_sapiens.GRCh38.dna.primary_assembly.fa \
-s -split -name \
-bed reference_transcriptome.bed \
> Homo_sapiens.GRCh38.98_transcriptome.fa

# Generate a fasta index
samtools faidx Homo_sapiens.GRCh38.98_transcriptome.fa

# Map
minimap2 -a -x map-ont -k14 --for-only \
Homo_sapiens.GRCh38.98_transcriptome.fa    basecalled.fastq > minimap_transcriptome.sam
```

Step 2 (recap): Map to the transcriptome

Required Input: (1) fastq files, (2) transcriptome in fasta format

```
# Convert the Ensembl GTF to BED
bedparse gtf2bed \
~/data/Homo_sapiens.GRCh38.102.chr.gtf \
> reference_transcriptome.bed

# Generate a Fasta from the GTF
bedtools getfasta \
-fi ~/data/Homo_sapiens.GRCh38.dna.primary_assembly.fa \
-s -split -name \
-bed reference_transcriptome.bed \
> Homo_sapiens.GRCh38.98_transcriptome.fa

# Generate a fasta index
samtools faidx Homo_sapiens.GRCh38.98_transcriptome.fa

# Map
minimap2 -a -x map-ont -k14 --for-only \
Homo_sapiens.GRCh38.98_transcriptome.fa    basecalled.fastq > minimap_transcriptome.sam
```

Expected output:

- Alignments in SAM format

Step 3 (recap): Filter alignments

Required Input: Alignments in SAM format

Step 3 (recap): Filter alignments

Required Input: Alignments in SAM format

```
# Filter, convert to BAM and sort
samtools view -b -F 2324 minimap_transcriptome.sam | samtools sort \
> minimap_transcriptome_filtered_sorted.bam

# Index
samtools index minimap_transcriptome_filtered_sorted.bam
```

Step 3 (recap): Filter alignments

Required Input: Alignments in SAM format

```
# Filter, convert to BAM and sort  
samtools view -b -F 2324 minimap_transcriptome.sam | samtools sort \  
    > minimap_transcriptome_filtered_sorted.bam  
  
# Index  
samtools index minimap_transcriptome_filtered_sorted.bam
```

Expected output:

- Filtered and sorted alignments in BAM format

Step 3 (recap): Filter alignments

Required Input: Alignments in SAM format

```
# Filter, convert to BAM and sort  
samtools view -b -F 2324 minimap_transcriptome.sam | samtools sort \  
    > minimap_transcriptome_filtered_sorted.bam  
  
# Index  
samtools index minimap_transcriptome_filtered_sorted.bam
```

Expected output:

- Filtered and sorted alignments in BAM format

Questions:

- What does 2324 mean?
- Would it be ok to use 2308?

Step 4 (recap): Resquiggle with Nanopolish

Required Input: (1) Fast5 files (2) Fastq (3) Alignments in BAM format (4) sequencing_summary.txt

Step 4 (recap): Resquiggle with Nanopolish

Required Input: (1) Fast5 files (2) Fastq (3) Alignments in BAM format (4) sequencing_summary.txt

```
#Index
> nanopolish index \
  -d ~/data/consortium_basecalled/workspace/ \
  -s ~/data/consortium_basecalled/sequencing_summary.txt \
  basecalled.fastq

# Run Nanopolish
> nanopolish eventalign \
  --scale-events --print-read-names --samples --signal-index \
  --reads basecalled.fastq \
  --bam mapped_reads.bam \
  --genome ~/data/Homo_sapiens.GRCh38.98_transcriptome.fa > eventalign.txt
[post-run summary] total reads: 21629, unparseable: 0, qc fail: 238, could not calibrate: 418, no
```

Step 5: Collapse eventalign files

Required Input: eventalign file (from Nanopolish)

Output: Collapsed file where all the events of each kmer are summarised (mean, median)

```
> nanocompare eventalign_collapse -h
usage: nanocompare eventalign_collapse [-h] [--eventalign EVENTALIGN] [--n_lines N_LINES]
                                         [--nthreads NTHREADS] [--outpath OUTPATH]
                                         [--outprefix OUTPREFIX] [--overwrite]
                                         [--log_level {warning,info,debug}] [--progress]

Collapse the nanopore eventalign output at kmers level and compute kmer level statistics
* Minimal example
    nanocompare eventalign_collapse -i nanopore_eventalign.tsv -outprefix out

Input options:
  --eventalign EVENTALIGN, -i EVENTALIGN
          Path to a nanopore eventalign tsv output file, or a list of file,
          or a regex (can be gzipped). It can be omitted if piped to standard
          input (default: piped to stdin)

Output options:
  --outpath OUTPATH, -o OUTPATH
          Path to the output folder (default: ./)
  --outprefix OUTPREFIX, -p OUTPREFIX
          text outprefix for all the files generated (default: out)
  --overwrite, -w
          Use --outpath even if it exists already (default: False)
```

Step 5: Collapse eventalign files

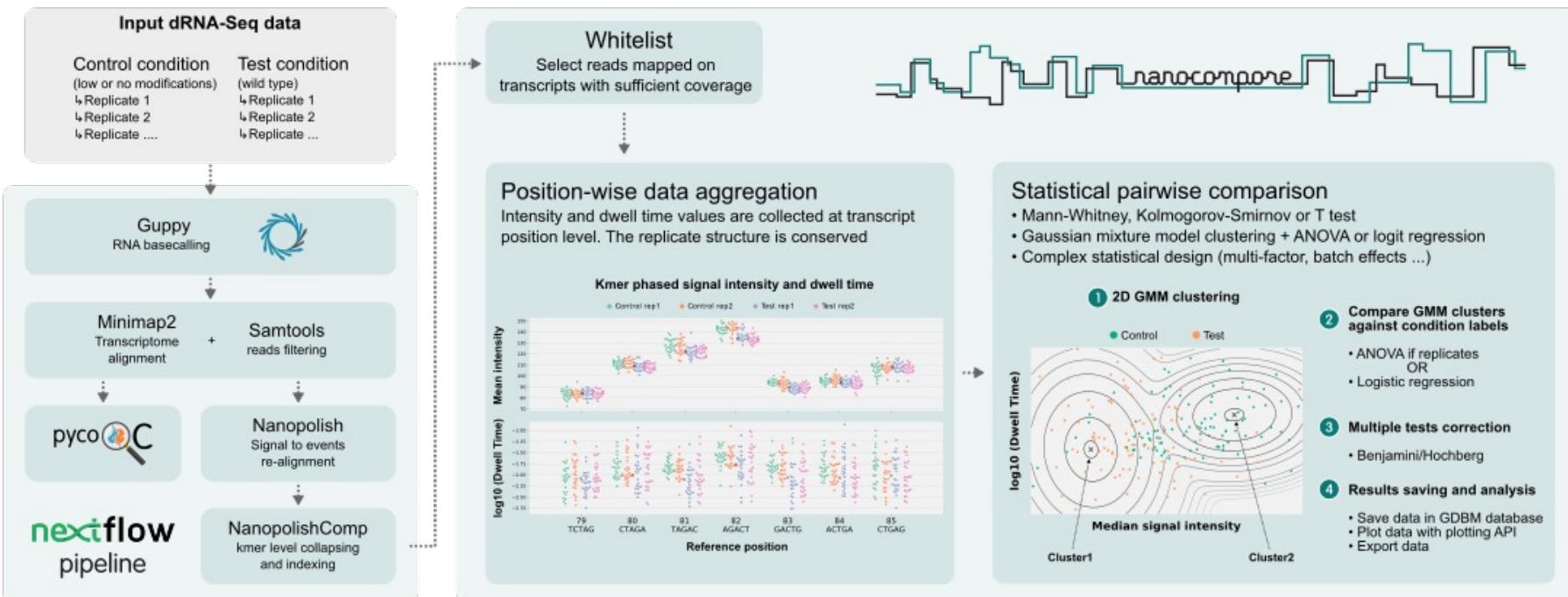
Input

```
> head eventalign.txt
contig      position  reference_kmer  read_name strand  event_index  event_level_mean  event
ENST00000451311(+) 22        AGACC       b69985a2    t        31          127.42           2.202
ENST00000451311(+) 22        AGACC       b69985a2    t        32          121.97           3.668
ENST00000451311(+) 22        AGACC       b69985a2    t        33          123.74           1.832
ENST00000451311(+) 22        AGACC       b69985a2    t        34          121.53           1.789
ENST00000451311(+) 22        AGACC       b69985a2    t        35          122.73           2.728
ENST00000451311(+) 22        AGACC       b69985a2    t        36          123.63           9.997
ENST00000451311(+) 23        GACCA       b69985a2    t        37          78.56            2.769
ENST00000451311(+) 23        GACCA       b69985a2    t        38          76.02            2.127
ENST00000451311(+) 24        ACCAG       b69985a2    t        39          71.39            2.717
```

Output

```
> head out_eventalign_collapse.tsv
b69985a2-7e7a-4f08-bd6d-ea7598d65b5c  ENST00000451311(+)
ref_pos  ref_kmer      num_events  num_signals  dwell_time      NNNNN_dwell_time  misma
22       AGACC         6           167          0.055449999403208494  0.0           0.0
23       GACCA         2           28           0.00929000019095838   0.0           0.0
24       ACCAG         1           24           0.007969999685883522  0.0           0.0
25       CCAGA         1           16           0.005309999920427799  0.0           0.0
26       CAGAC         5           128          0.042489999905228615  0.0           0.0
27       AGACT         5           139          0.04614999983459711  0.0           0.0
28       GACTT         2           26           0.008630000287666917  0.0           0.0
29       ACTTC         1           7            0.002319999970495701  0.0           0.0
```

Step 6: Run Nanocompore



Step 6: Run Nanocompore

Required Input:

- eventalign_collapse files (one per sample)
- Transcriptome Fasta
- Transcriptome BED (optional)

```
> nanocompore sampcomp \
    --file_list1 WT_rep1_ealign_collapsed.tsv,WT_rep2_ealign_collapsed.tsv
    --file_list2 KD_rep1_ealign_collapsed.tsv,KD_rep2_ealign_collapsed.tsv
    --label1 WT \
    --label2 KD \
    --fasta reference_transcriptome.fa \
    --bed reference_transcriptome.bed \
    --outpath ~/data/nanocompore_METTL3_KD/ \
    --sequence_context 2 \
    --allow_warnings \
    --pvalue_thr 0.01 \
    --min_coverage 30 \
    --logit \
    --nthreads 3
```

Nanocompore output

```
> ls -lah ~/data/nanocompore_METTL3_KD/
drwxr-xr-x 2 ubuntu ubuntu 4.0K Jan 20 13:18 .
drwxr-xr-x 5 ubuntu ubuntu 4.0K Jan 20 13:16 ..
-rw-rw-r-- 1 ubuntu ubuntu 40G Jan 20 13:18 out_SampComp.db
-rw-r--r-- 1 ubuntu ubuntu 172M Jan 20 11:56 out_nanocompore_results.tsv
-rw-r--r-- 1 ubuntu ubuntu 154M Jan 20 11:56 out_nanocompore_shift_stats.tsv
-rw-r--r-- 1 ubuntu ubuntu 1.3M Jan 20 11:56 out_sig_sites_GMM_logit_pvalue_context_2_thr_0.05.
-rw-r--r-- 1 ubuntu ubuntu 26M Jan 20 11:56 out_sig_sites_GMM_logit_pvalue_context_2_thr_0.05.
-rw-r--r-- 1 ubuntu ubuntu 723K Jan 20 11:56 out_sig_sites_GMM_logit_pvalue_thr_0.05.bed
-rw-r--r-- 1 ubuntu ubuntu 26M Jan 20 11:56 out_sig_sites_GMM_logit_pvalue_thr_0.05.bedgraph
```

Nanocompore output

```
> ls -lah ~/data/nanocompore_METTL3_KD/
drwxr-xr-x 2 ubuntu ubuntu 4.0K Jan 20 13:18 .
drwxr-xr-x 5 ubuntu ubuntu 4.0K Jan 20 13:16 ..
-rw-rw-r-- 1 ubuntu ubuntu 40G Jan 20 13:18 out_SampComp.db
-rw-r--r-- 1 ubuntu ubuntu 172M Jan 20 11:56 out_nanocompore_results.tsv
-rw-r--r-- 1 ubuntu ubuntu 154M Jan 20 11:56 out_nanocompore_shift_stats.tsv
-rw-r--r-- 1 ubuntu ubuntu 1.3M Jan 20 11:56 out_sig_sites_GMM_logit_pvalue_context_2_thr_0.05.
-rw-r--r-- 1 ubuntu ubuntu 26M Jan 20 11:56 out_sig_sites_GMM_logit_pvalue_context_2_thr_0.05.
-rw-r--r-- 1 ubuntu ubuntu 723K Jan 20 11:56 out_sig_sites_GMM_logit_pvalue_thr_0.05.bed
-rw-r--r-- 1 ubuntu ubuntu 26M Jan 20 11:56 out_sig_sites_GMM_logit_pvalue_thr_0.05.bedgraph
```

Main results table

```
> head ~/data/nanocompore_METTL3_KD/out_nanocompore_results.tsv
pos chr genomicPos ref_id strand ref_kmer GMM_anova_pvalue GMM_anova_pvalue_context
1417 1 8861365 ENSTooooo647408 - CAAGT 1.0 0.999999999998306
1422 1 8861360 ENSTooooo647408 - AAGCT 1.0 0.999999999998306
1424 1 8861358 ENSTooooo647408 - GCTGT 1.0 0.999999999998306
1425 1 8861357 ENSTooooo647408 - CTGTG nan nan
1426 1 8861356 ENSTooooo647408 - TGTGG 1.0 0.999999999998306
1427 1 8861355 ENSTooooo647408 - GTGGG 1.0 0.999999999998306
1429 1 8861353 ENSTooooo647408 - GGGCA 1.0 0.999999999998306
1430 1 8861352 ENSTooooo647408 - GGCAG 1.0 0.999999999998306
1440 1 8861342 ENSTooooo647408 - CCCTT 1.0 0.999999999998306
```

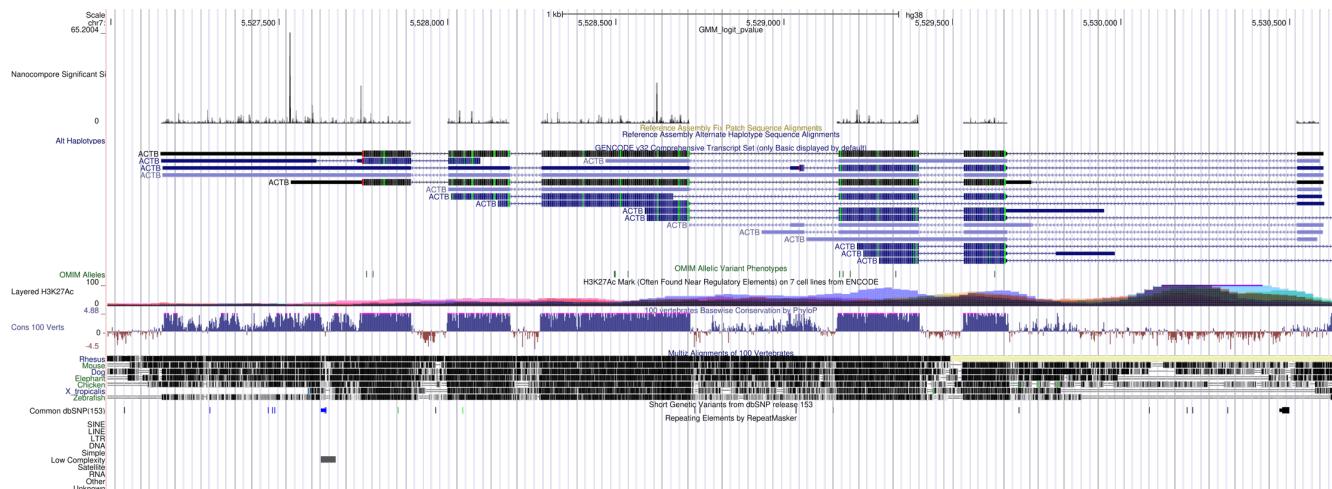
Nanocompare BED tracks

Nanocompare also creates BED and BEDGRAPH tracks of significant hits.

Download the file:

`out_sig_sites_GMM_logit_pvalue_thr_0.05.bedgraph`

and load it into the UCSC genome browser.



Visualise Nanocompore results

```
library(tidyverse)

NANOCOMPORE <- "~/data/nanocompore_METTL3_KD/out_nanocompore_results.tsv"

nanocompore <- read_tsv(NANOCOMPORE, col_types="icicccdddddcciccc") %>%
  mutate(TxId=gsub(":::+", "", ref_id))

drach <- c("AAACA", "AAACC", "AAACT", "AGACA", "AGACC", "AGACT", "GAACA", "GAACC", "GAACT",
          "GGACA", "GGACC", "GGACT", "TAACA", "TAACC", "TAACT", "TGACA", "TGACC", "TGACT")

volcano_abs <-
  nanocompore %>%
  filter(Logit_LOR!="NC", !is.na(GMM_logit_pvalue), GMM_logit_pvalue<0.1) %>%
  mutate(LOR=as.numeric(Logit_LOR)) %>%
{
  ggplot(., aes(x=abs(LOR), y=-log10(GMM_logit_pvalue))) +
    geom_point(alpha=0.8, aes(colour=ref_kmer %in% drach)) +
    ggrepel::geom_text_repel(data=top_n(., 20, -log10(GMM_logit_pvalue)), size=4, aes(label=paste0
      xlab("Logistic regression\nodds ratio") +
      ylab("Nanocompore p-value (-log10)") +
      scale_colour_discrete(name="DRACH motif") +
      theme_bw(18)
}

ggsave(volcano_abs, file="sharkfin_plot.pdf", width=12, height=10)
```

The sharkfin plot

