**Data analysis pipeline**

The software here has been tweaked and updated slightly to cope with some anomalous behaviours that sometimes arose and to add additional outputs.

Normally we map nanopore data to the target virus genome using minimap2, here is the setup we used for a DNA virus like adenovirus:

minimap2 -a -x splice -uf -k13 --sam-hit-only target\_genome.fasta nanopore\_reads.fastq > output.sam

But we also use different minimap2 parameters (please see the manual page of minimap2 for more detail) depending on the circumstances of the virus – e.g. for RSV we use:

minimap2 -a -x map-ont -un -k13 --sam-hit-only target\_genome.fasta nanopore\_reads.fastq > output.sam

The first step is to use that sam file and the genome with “classify\_transcripts\_and\_polya\_segmented\_V2.pl” here is a typical command line:

***classify\_transcripts\_and\_polya\_segmented\_V2.pl prefix polya/TSS\_window splice\_window polya\_min nanopolish\_polya.txt input.sam genome.fasta 1 100***

**prefix**

*Enter a short prefix that will help you remember what the outputs were for!*

**polya/TSS\_window**

*Enter an integer for the size of window you want the software to use to bundle the transcription start or stop sites together (we used 15 in our analysis)*

**splice\_window**

*Similarly for bundling splice sites together, normally not needed if the nanopore data has been corrected by illumina but can be useful if using uncorrected data and you have sufficient depth of read that a “true” splice site will rise above the noise.*

**polya\_min**

*If you have polya.txt files from the nanopolish polya utility then you can specify here the minimum length of polya required alongside the “PASS” flag from nanopolish polya before a transcript will be analysed.*

**Nanopolish\_polya.txt**

*If you have polya.txt files from the nanopolish polya utility then put the name of the files here, if you don’t have this data the just put no\_polya instead and the script will carry on without that data and just put a poly A length of -1 for everything.*

**Input.sam**

*Here you put the name of your sorted sam file.*

**Genome.fasta**

*Here you put the name of the file that contains the genome you mapped your data to*

**1**

*This is the minimum copy number of transcript group you want the software to consider, we used 1*

**100**

*This is the maximum number of entries you want to see in the gff file called “Prefix.gff\_most\_popular\_list.gff3” file, this GFF3 file is useful in the early analysis steps to give you a broad overview of what is being made as it will return (in this example) only the top 100 most abundant transcript groups found.*

The software will examine each transcript and note where the start and end locations of the mapped transcript along with the locations of splice acceptor and donor sites and the amount of soft clipping. This is used to group transcripts together if they have start and end locations within the window specified and if they have the same splice pattern. For example, once the start locations of all the transcripts are known the software then ranks each start location based on how many times it is used. The most used transcription start site is selected first and all other start sites for within your specified window are corrected so that they now all start at that location. Thus, all the transcript groups identified with a start location within that window will have their transcription start sites modified to reflect the dominant transcription start site. The outputs are:

**prefix.canonical\_transcripts\_by\_abundance.fasta**

*A list of pseudo transcripts generated by using the start, end and splice patterns of the transcript groups found, one transcript per transcript group along with information on how many individual reads belong to that transcript group, the average polyA length (if available) and the mapping co-ordinates.*

**prefix.GFF\_all\_found.gff3**

*A gff3 file describing each transcript group for visualisation in a viewer (e.g. IGV viewer).*

**prefix.GFF\_most\_popular\_list.gff3**

*As above but only listing the 100 most abundant transcript groups.*

**prefix.raw\_soft\_clip\_locations.txt**

*A tab separated list of the locations on the genome where there is soft clipping on either the plus or minus strand*

**prefix.start\_sad\_stop\_pattern\_count.txt**

*A tab separated table listing all the transcript groups describing each groups start location, end location, strand,* ***s****plice* ***a****cceptor/****d****onor locations (referred to as the* ***sad*** *location) how many transcripts belong to that group and the average and standard deviation for the poly A length (where available). This file is used by the next software stepto analyse the genome and assign features and ORFS to the transcript groups.*

**prefix.raw\_processed\_start\_sad\_polya.txt**

*Tab separated list describing for each nucleotide position how often that is the location of a transcript start, a transcript end or a* ***s****plice* ***a****cceptor/****d****onor. Both the raw counted numbers are listed alongside the cumulative counts of events at each nucleotide location* ***after*** *the software has grouped nearby events together using the window size specified in the command line.*

Using **name\_transcripts\_and\_track\_ssc.pl** to find ORFs and determine what features are present on each transcript as defined by the transcript grouping software is the next step.

The command line is:

**name\_transcripts\_and\_track\_ssc\_V2.pl prefix ssc\_nt features\_table.txt genome.fasta start\_sad\_stop\_pattern\_count.txt**

**prefix**

*Enter a short prefix that will help you remember what the outputs were for!*

**ssc\_nt**

*Enter a value for* ***s****tart* ***s****ite* ***c****orrection – the number of upstream nucleotides you want the pseudo transcript to include in order to account for the loss of 5’ nucleotides in nanopore sequencing. We used a value of 10.*

**features\_table.txt**

*The name of the text file containing a list of features on the genome you are analysing (see below for format).*

**genome.fasta**

*The name of the fasta file with the viral genome.*

**start\_sad\_stop\_pattern\_count.txt**

*The name of the file from the first script that contains the information on the transcript groups*

For this script to work, the user supplies a “features\_table.txt” list of features, their locations on the viral genome and the name you want associated with each feature – conceptually similar to a GFF3 file but simpler. For example in each tab field in order you provide: the genome name<TAB>strand<TAB>location of the start codon<TAB>sad location<SPACE>sad location<… repeat as needed or leave blank if not needed…><TAB>location of stop codon<TAB> name of the feature.

Here is the entry for the ORF for the E1a 12S protein:

gi|56160529|ref|AC\_000008.1| + 559 975 1229 1546 E1a\_12S

This tells the software that between 559 and 975 there is an exon and between 1229 and 1546 there is an exon, the two should be connected and the transcript covered by these regions contains a feature called E1a\_12S.

When adding a free text name to the feature please use underscore instead of space.

When describing features on the opposite strand, the entries for start and stop will be the actual locations of the start and stop, but the sad locations describing the exon/intron boundaries must be in numerical order:

gi|56160529|ref|AC\_000008.1| - 14120 10590 14111 8583 preTP(E2B)

If you add the text “\_TSS” then the software will understand that you are describing a transcription start site and any transcript group that has a start located within 50nt of that location will have this feature description added to the “features found” part of the output (see below).

This file is used by the software to generate pseudo transcripts based on the supplied coordinates. These pseudo transcripts are stored for comparison with your transcript groups later and they are also scanned 5’ - 3’ to determine if there is an AUG and if the sequence of the protein is 6aa or longer then it too is stored. If the ORF is stored then it is listed as a canonical ORF, a fasta file of the ORFS it thinks it has found in this way and the names associated with them is produced. Next the software examines each transcript group in the same way, generating a pseudo transcript that is then examined to see if it contains any of the nucleotide sequences described by the features file. These features found and the splice acceptor/donor sequences at any intron/exon boundaries are also noted and reported. Allied to this the software then scans from 5’ to 3’ on the pseudo transcript looking for the first and second AUGs, both are translated and if the first AUG is the start codon for a canonical ORF then that is listed as the primary ORF. If the first AUG does not lead to a canonical ORF then the second ORF is examined to see if that one is the beginning of a canonical ORF. If neither are canonical ORFs then the primary ORF is reported alongside the transcript group and the transcript group is flagged as having no known ORFs and the transcript itself is added to the tab separated line of information about that transcript group.

The following files are produced:

**prefix.TSS\_start\_sad\_polyA\_count\_list.txt**

*This is a list (in descending order of number of observed transcripts belonging to each group) of all the transcript groups analysed.*

**prefix.proteins\_known.fasta**

*A fasta file of the proteins the software found after analysing the canonical features list you provided.*

**prefix.proteins\_not\_known.fasta**

*A fasta file of proteins the software found after analysing the transcript groups that could not be matched to the list of canonical proteins.*

**prefix.most\_abundant\_trans\_per\_orf.gff**

*A GFF3 format file describing the most abundant transcript group that will code for each one of the canonical ORFs.*

**prefix.count\_of\_translated\_features.txt**

*A tab delimited file detailing, for each ORF identified, how many transcripts in total would code for that ORF and (if available) the average poly A length for the dominant transcript group that codes for the indicated ORF.*

**prefix.combined\_counts\_and\_feature\_names.txt**

*A tab delimited file containing a suumary for each transcript group. The strand, start, sad locations and poly adenylation site alongside the number of observed transcripts belonging to that group. In addition, a unique name for each transcript group, the average polyA length and standard deviation, a list of features found on that transcript, the first ORF found, the second ORF found (if the first AUG does not code for a canonical ORF), the sequence of the first ORF and the sequence of the transcript if the transcript does not contain any recognised ORFs.*

**prefix.splice\_acceptor\_donor\_usage.txt**

A list of splice acceptor donor pair usage (in message sense only) and how often each one is used.

**prefix (folder)**

*The software will also create a folder with the prefix used as the name for the folder. Inside the folder is a series of GFF3 files, one for each ORF found. Thus the user can examine the GFF3 file for, say, the hexon protein and gain an overview of what transcript groups appear to code for that protein.*

When the software tries to identify a canonical ORF on a transcript it will also look at some simple alternatives if the first AUG dies not code for a canonical ORF:

If the only canonical ORF found is initiated from the second methionine then the text “\_2nd\_M” will be added to the name of the ORF found.

If the ORF found contains a canonical ORF but the sequence has been extended/truncated then the text “possible 5’(or 3’) truncation (or extension) of…” will be inserted into the description.

**Summary of data processing steps when you also have matched illumina data**

1. Obtain nanopore reads (we removed the reads mapping to human genes to reduce the size of the data and reduce compute times).
2. Obtain matched illumina data
3. Map illumina data to the adenovirus (or your organism’s) genome and separate virus mapping reads from non-mapping reads then convert the mapping reads back to fastq data.
4. (OPTIONAL) use Trinity read normalisation to reduce the size of the illumina fastq files.
5. Use LorDEC to correct the nanopore data using the illumina data – we used repeated iterative rounds with increasing K-mer values. Any software that does this is fine to use.
6. Generate a features file as we have described for adenovirus for your organism (making sure the genome name match the SAM file is critical). The exact location of the starts and stops is not too important as the software will translate so you can use start and stop locations a few nucleotides either side if you want to.
7. Us the two perl scripts to find featires/ORF and characterise your data.

**Comparing Nanopore splice sites to Illumina splice sites**

For this analysis we used a SAM file of illumina reads that were mapped to the adenovirus genome and we compared it to the list of identified splicing events as listed in the **prefix.combined\_counts\_and\_feature\_names.txt** file produced by the pipeline above. The software included in the supplementary data is called compare\_nanopore\_splices\_to\_illumina.pl and it also requires a fasta file of the adenovirus genome. This software first analyses the illumina SAM file and identified splicing events and uses the viral genome to identify the splice acceptor/donor sites along with their location. This data is then compared to the list of splice sites present in the **prefix.combined\_counts\_and\_feature\_names.txt** file.

The command line is:

***compare\_nanopore\_splices\_to\_illumina.pl prefix\_2 mapped\_illumina\_reads.sam prefix.combined\_counts\_and\_feature\_names.txt genome.fasta***

**prefix**

*Enter a short prefix that will help you remember what the outputs were for!*

**mapped\_illumina\_reads.sam**

*This is the sam file format of your reads mapped to the target genome*

**prefix.combined\_counts\_and\_feature\_names.txt**

*Provide the counts and feature names text file generated in the earlier analysis of the nanopore data.*

**genome.fasta**

*A fasta file of the genome used to map the illumina data.*

Three files are produced:

**Prefix.illumina\_juncs\_and\_summary.txt**

This file lists the locations and splice acceptor donor pairs of all the splice events found in the illumina data and it also lists the *distinct* mapping sites for every illumina read that contains the names splice event, how many distinct start mapping sites were observed plus the total depth of reads that cover the named splice event.

**Prefix.evidence\_good.txt**

This is a file that is in format the same as the **prefix.combined\_counts\_and\_feature\_names.txt** file but here are all the transcripts for which every splice event has evidence for it in the illumina file, the features column will have additional information added in the name of **FIIR** and **RIIIR** which stand for **F**orward **I**ndependent **I**llumina **R**eads and **R**everse **I**ndependent **I**llumina **R**eads. This is a count of how many reads cover the splice site that have unique mapped start sites and whether they are mapped to the forward or reverse strand.

**Prefix.evidence\_poor.txt**

This is the same as above but containing all the transcripts that have at least one splice event that is not present in the illumina data. In addition, if the splice sites do not have canonical splice acceptor donor pairs (GU-AG, GC-AG, AT-AC or GT-GG) this will be flagged here.