Stranded Mapping from Long Reads

David Eccles

Abstract

This protocol demonstrates how to convert raw long reads produced using a strand-specific sequencing protocol (e.g. ONT's strand-switching protocol) into strand-specific mapped reads.

The general ides is to use LAST to identify the adapter orientation relative to the genome, and then use that information to split BAM files up and recombine them to create two strand-specific files that are displayable in a genome browser.

Citation: David Eccles Stranded Mapping from Long Reads. **protocols.io**
dx.doi.org/10.17504/protocols.io.n8cdhsw
Published: 04 Apr 2018

Before start

I have written my own script to process LAST results into a CSV format, which makes it easier to do line-by-line data filtering. I have also created a fastq filtering script that helps for filtering reads into different files.

You will also need access to the following free and open-source software programs:

- LAST
- minimap2
- samtools

And the following additional data files:

- a FASTA file containing strand-specific primer / adapter sequences.

Protocol

Read Correction

Step 1.

I prefer starting off my data analysis with a read correction with Canu (ideally v1.7 now that it's out, because that attempts correction of *all* reads, but here I use canu v1.6). I use `minimap` as the mapper to speed this up. The `genomeSize` parameter should be approximately a tenth to a fortieth of the number of bases in your dataset to make sure that no sequences are excluded (bigger is better, as long as Canu doesn't freak out about memory consumption):

```
/install/canu/canu-1.6/Linux-amd64/bin/canu overlapper=minimap \
  genomeSize=100M minReadLength=100 minOverlapLength=30 -correct \
  -p 4T1_BC06 -d 4T1_BC06 -nanopore-raw \
  workspace/pass/barcode06/fastq_runid_*.fastq
```

This creates a file `4T1_BC06/4T1_BC06.correctedReads.fasta.gz`.

Chimeric read filtering

Step 2.

The next step I carry out is a basic read-level QC to exclude chimeric reads. Porechop can be used for this, although that removes adapters by default, which is not particularly useful in this case.

I use LAST to search for adapter sequences within the corrected reads, pass it through my conversion script, and extract out duplicated mappings (i.e. where the same read/adapter pair appears more than once in the mapping results):

```
lastal -P 10 ONT_barcodes_adapters.fa <(zcat
4T1_BC06/4T1_BC06.correctedReads.fasta.gz) | \
  /scripts/maf_bcsplit.pl | awk -F',' '{print $1,$2}' | sort | \
  uniq -d | awk '{print $1}' | uniq > reads_with_duplicated_adapters.txt
```

I use this file to filter out chimeric reads from the corrected dataset using another fastq filtering script I've created:

```
/scripts/fastx-fetch.pl -v -i reads_with_duplicated_adapters.txt
4T1_BC06/4T1_BC06.correctedReads.fasta.gz | \
  gzip > 4T1_BC06.correctedReads.uniqueOnly.fasta.gz
```

Adapter filtering

Step 3.

Along roughly similar lines to the chimeric read filtering, I then look for the strand switch and VNT adapters in the sequences. For a forward-orientated query, I expect the strand switch primer to be in the forward direction, and the VNP primer to be in the reverse direction:

```
lastal -P 10 ONT_barcodes_adapters.fa <(zcat
4T1_BC06.correctedReads.uniqueOnly.fasta.gz) | \
  /scripts/maf_bcsplit.pl | grep -e 'ONT_SSP,+' -e 'ONT_VNP,-' | \
  awk -F',' '{print $1}' | sort | uniq > fwdQry_seqs_BC06.txt
lastal -P 10 barcodes_primerSeqs.fa <(zcat
4T1_BC06.correctedReads.uniqueOnly.fasta.gz) | \
  /scripts/maf_bcsplit.pl | grep -e 'ONT_SSP,-' -e 'ONT_VNP,+' | \
  awk -F',' '{print $1}' | sort | uniq > revQry_seqs_BC06.txt
```

And then more filtering to split the reads up into forward and reverse [genome-relative] subsets:

```
/scripts/fastx-fetch.pl -i fwdQry_seqs_BC06.txt
```

```
4T1_BC06.correctedReads.uniqueOnly.fasta.gz | \
  gzip > fwd_4T1_BC06.correctedReads.uniqueOnly.fasta.gz
/scripts/fastx-fetch.pl -i revQry_seqs_BC06.txt
4T1_BC06.correctedReads.uniqueOnly.fasta.gz | \
  gzip > rev_4T1_BC06.correctedReads.uniqueOnly.fasta.gz
```

Read Mapping

Step 4.

Now that the reads have been oriented, the mapping can be done:

```
/install/minimap2/minimap2 -t 10 -a mmus_ucsc_all_cdna.idx -x splice \
  fwd_4T1_BC06.correctedReads.uniqueOnly.fasta.gz \
  > fwd_4T1_BC06.CU_vs_mmus.bam
/install/minimap2/minimap2 -t 10 -a mmus_ucsc_all_cdna.idx -x splice \
  rev_4T1_BC06.correctedReads.uniqueOnly.fasta.gz \
  > rev_4T1_BC06.CU_vs_mmus.bam
```

Splitting and recombining strands

Step 5.

The mapped files are then split based on their mapping direction using `samtools view` flag filtering to exclude or include reverse-mapped reads:

```
samtools view -F 0x10 -b fwd_4T1_BC06.CU_vs_mmus.bam >
fwd_fwd_4T1_BC06.CU_vs_mmus.bam
samtools view -f 0x10 -b fwd_4T1_BC06.CU_vs_mmus.bam >
rev_fwd_4T1_BC06.CU_vs_mmus.bam
samtools view -F 0x10 -b rev_4T1_BC06.CU_vs_mmus.bam >
fwd_rev_4T1_BC06.CU_vs_mmus.bam
samtools view -f 0x10 -b rev_4T1_BC06.CU_vs_mmus.bam >
rev_rev_4T1_BC06.CU_vs_mmus.bam
```

And finally the files are recombined to identify forward and reverse-encoded transcripts via `samtools merge`. I'm calling the "pos" strand the one that is encoded in the same direction as the genome, and the "neg" strand the reverse-complement direction:

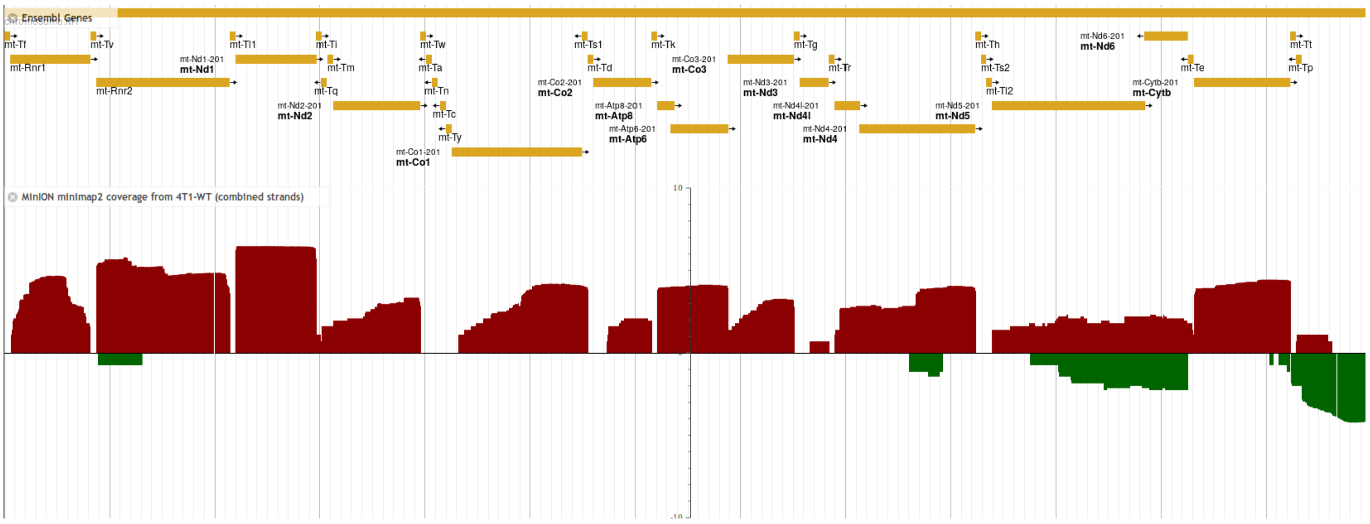
```
samtools merge pos_4T1_BC06.CU_vs_mmus.bam \
  rev_fwd_4T1_BC06.CU_vs_mmus.bam fwd_rev_4T1_BC06.CU_vs_mmus.bam
samtools merge neg_4T1_BC06.CU_vs_mmus.bam \
  fwd_fwd_4T1_BC06.CU_vs_mmus.bam rev_rev_4T1_BC06.CU_vs_mmus.bam
```

Alternatively, it's possible to set the BAM flags for read 1 and read 2 for the reverse and forward-encoded adapters respectively, which allows everything to then be combined into a single BAM file (and treated in the same way as a strand-specific Illumina BAM file).

Sanity Check

Step 6.

If this has worked properly, then mapping human or mouse to the mitochondrial genome should show most expression appearing on the positive strand, with a small scattering of negative-strand expression, a bit like this:



Warnings

These scripts have been slightly modified from scripts that I have run. Consider them demonstrative: pay attention to the words rather than the script.