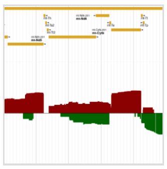


Version 6

Oct 19, 2020

Stranded Mapping from Oriented Long Reads V.6

David A Eccles[1]

[1]Malaghan Institute of Medical Research (NZ)

1 *Works for me* dx.doi.org/10.17504/protocols.io.bnjrmcm6

David Eccles
Malaghan Institute of Medical Research (NZ)

ABSTRACT

This protocol demonstrates how to map strand-oriented long reads to a genome, and visualise them in a genome browser.

The general idea is to use minimap2 to create stranded BAM files, which are split for forward/reverse orientation then converted into BigWig format for display in a genome browser.

Input(s):

- stranded fastq files (see protocol Preparing Reads for Stranded Mapping)
- a FASTA file containing the genome / sequence of interest.

Output(s):

- Genome-mapped stranded BAM files
- Genome-mapped stranded BigWig files

EXTERNAL LINK

https://bioinformatics.stackexchange.com/a/3922/73

DOI

dx.doi.org/10.17504/protocols.io.bnjrmcm6

EXTERNAL LINK

https://bioinformatics.stackexchange.com/a/3922/73

PROTOCOL CITATION

David A Eccles 2020. Stranded Mapping from Oriented Long Reads. **protocols.io**
https://dx.doi.org/10.17504/protocols.io.bnjrmcm6
Version created by David Eccles

EXTERNAL LINK

https://bioinformatics.stackexchange.com/a/3922/73

CREATED

Oct 19, 2020

LAST MODIFIED

Oct 19, 2020

PROTOCOL INTEGER ID

43345

ABSTRACT

This protocol demonstrates how to map strand-oriented long reads to a genome, and visualise them in a genome browser.

The general idea is to use minimap2 to create stranded BAM files, which are split for forward/reverse orientation then converted into BigWig format for display in a genome browser.

Input(s):
- stranded fastq files (see protocol Preparing Reads for Stranded Mapping)
- a FASTA file containing the genome / sequence of interest.

Output(s):
- Genome-mapped stranded BAM files
- Genome-mapped stranded BigWig files

BEFORE STARTING

You will need access to the following free and open-source software program(s):
- minimap2
- samtools

And the following additional data file(s):
- a FASTA file containing the genome / sequence of interest.

Orient Reads

1 Orient reads as per protocol Preparing Reads for Stranded Mapping.

If this has been done, then the following command should produce output without errors:

```
for bc in $(awk '{print $2}' barcode_counts.txt);
  do ls oriented/${bc}_reads_dirAdjusted.fq.gz;
done
```

Example output:

```
oriented/BC03_reads_dirAdjusted.fq.gz
oriented/BC04_reads_dirAdjusted.fq.gz
oriented/BC05_reads_dirAdjusted.fq.gz
oriented/BC06_reads_dirAdjusted.fq.gz
oriented/BC07_reads_dirAdjusted.fq.gz
oriented/BC08_reads_dirAdjusted.fq.gz
```

Index Preparation

2 Prepare genome index for spliced alignment

minimap2 2.16
Linux
source by Heng Li

```
minimap2 -d mmus_ucsc_all-splice.idx -Q -t 10 -x splice mmus_ucsc_all.fa
```

Read Mapping

3 Map the long reads to the genome using minimap2, using samtools to covert to a sorted BAM format. This is where the reverse complementing done during demultiplexing gives a big saving of effort. As this BAM file is one of the main outputs, the run name is added to the file name.

SAMtools 1.8
Linux
source by Wellcome Trust Sanger Institute

```
runName="CHANGE_THIS";
mkdir -p mapped;
for bc in $(awk '{print $2}' barcode_counts.txt);
  do echo ${bc};
  minimap2 -t 10 -a -x splice mmus_ucsc_all-splice.idx
oriented/${bc}_reads_dirAdjusted.fq.gz | \
    samtools view -b | samtools sort > mapped/mm2_${runName}_called_${bc}_vs_MmusG.bam;
done
```

Creating BigWig Coverage Files

4 **mpileupDC.pl**

A bedGraph of coverage is created using samtools mpileup and mpileupDC.pl, excluding any skipped intronic sequence. When 'mpileupDC.pl' is provided with a single file, it will output a bedGraph file with a header line starting with '##'; this header line is removed.

To simplify output naming in later steps, the run name is added to the file name.

The particular JBrowse plugin that I use for stranded display requires that the reverse strand have *negative* coverage values, so that file needs to be changed:

```
runName="CHANGE_THIS";
for bc in $(awk '{print $2}' barcode_counts.txt);
  do echo ${bc};
  samtools view -b -F 0x10 mapped/mm2_called_${bc}_vs_MmusG.bam | \
    samtools mpileup -A -B -Q 0 -q 0 -I -q 0 -Q 0 - | \
    mpileupDC.pl | tail -n +2 > mapped/mm2_${runName}_called_${bc}_vs_MmusG.bg.plus
  samtools view -b -f 0x10 mapped/mm2_called_${bc}_vs_MmusG.bam | \
    samtools mpileup -A -B -Q 0 -q 0 -I -q 0 -Q 0 - | \
    mpileupDC.pl | tail -n +2 > mapped/mm2_${runName}_called_${bc}_vs_MmusG.bg.minus
  perl -i -pe 's/([0-9]+)$/-$1/' mapped/mm2_${runName}_called_${bc}_vs_MmusG.bg.minus
done;
```

As an alternative, BEDTools can be used to generate coverage. The default options for BEDTools treat sequence deletions (which happen frequently in nanopore reads) as a drop in coverage, which can make exon hunting and coverage calculation more difficult. I have proposed a fix to this via a command-line option "-ignoreD", which is available from my BEDTools fork:

BEDTools github-ignoreD
Linux
source by Aaron Quinlan

```
runName="CHANGE_THIS";
for bc in $(awk '{print $2}' barcode_counts.txt);
  do echo ${bc};
  samtools view -q 1 mapped/mm2_called_${bc}_vs_MmusG.bam | \
    bedtools genomecov -bga -strand '+' -split -ignoreD -ibam - >
mapped/mm2_${runName}_called_${bc}_vs_MmusG.bg.plus
    bedtools genomecov -bga -strand '-' -split -ignoreD -ibam - >
mapped/mm2_${runName}_called_${bc}_vs_MmusG.bg.minus
  perl -i -pe 's/([0-9]+)$/-$1/' mapped/mm2_${runName}_called_${bc}_vs_MmusG.bg.minus
done;
```

With the standard version of BEDTools, the "-ignoreD" parameter can be excluded in order to generate a similar BedGraph output, but with drops in coverage at deletion points:

```
runName="CHANGE_THIS";
for bc in $(awk '{print $2}' barcode_counts.txt);
  do echo ${bc};
  samtools view -q 1 mapped/mm2_called_${bc}_vs_MmusG.bam | \
    bedtools genomecov -bga -strand '+' -split -ibam - >
mapped/mm2_${runName}_called_${bc}_vs_MmusG.bg.plus
    bedtools genomecov -bga -strand '-' -split -ibam - >
mapped/mm2_${runName}_called_${bc}_vs_MmusG.bg.minus
  perl -i -pe 's/([0-9]+)$/-$1/' mapped/mm2_${runName}_called_${bc}_vs_MmusG.bg.minus
done;
```

5 Stranded bedgraph files are converted to bigwig. This requires BEDTools and a genome information file containing chromosome lengths (one for Mmus/mm10 is attached to this step). As this bigwig file is one of the main outputs, the run name is added to the file name.

BEDTools 2.26.0
source by Quinlan laboratory, University of Utah

```
for bc in $(awk '{print $2}' barcode_counts.txt);
  do echo ${bc};
  basename="mapped/mm2_${runName}_called_${bc}_vs_MmusG"
  bedGraphToBigWig ${basename}.bg.plus Mmus_genome.chrInfo.txt ${basename}.bw.plus
  bedGraphToBigWig ${basename}.bg.minus Mmus_genome.chrInfo.txt ${basename}.bw.minus
done
```

Mmus_genome.chrInfo.txt

JBrowse Configuration

6 Each track should have its own JBrowse configuration section using the *StrangedBigWig* class and *StrandedXYPlot* type. BAM tracks can also be added. An example is shown here:

```
[tracks.bam-CG005_Nov18_BC03-track]
# settings for what data is shown in the track
```

```
storeClass     = JBrowse/Store/SeqFeature/BAM
urlTemplate    = raw/mm2_called_CG005AB_BC03_vs_MmusG.bam
baiUrlTemplate = raw/mm2_called_CG005AB_BC03_vs_MmusG.bam.bai
chunkSizeLimit = 10000000
maxHeight      = 3000
# settings for how the track looks
category = MinION / Alignments
type = JBrowse/View/Track/Alignments2
key  = Minimap2 alignments from 4T1.p0#C [CG005]

[tracks.bw-CG005_Nov18_BC03-both-track]
storeClass     = StrandedPlotPlugin/Store/SeqFeature/StrandedBigWig
urlTemplate    = bw/mm2_called_CG005AB_BC03_vs_MmusG.bw
category       = MinION / Coverage
type           = StrandedPlotPlugin/View/Track/Wiggle/StrandedXYPlot
key            = Minimap2 coverage from 4T1.p0#C [CG005]
scale          = log
scoreType      = maxScore
autoscale      = global
style.pos_color = #228B22
style.neg_color = lightskyblue
```

I have written my own helper scripts to reduce the effort required to generate these track sections. This is specific to my use case, but may help others for adapting to their own use. It takes three command-line parameters: the base file name, the internal ID for use in the track definition, and the label that appears within JBrowse. I use it as follows:

```
./makeMinIONTemplate.sh mm2_called_CG005AB_BC03_vs_MmusG CG005_Nov18_BC03 '4T1.p0#C
[CG005]' >> tracks.conf
```

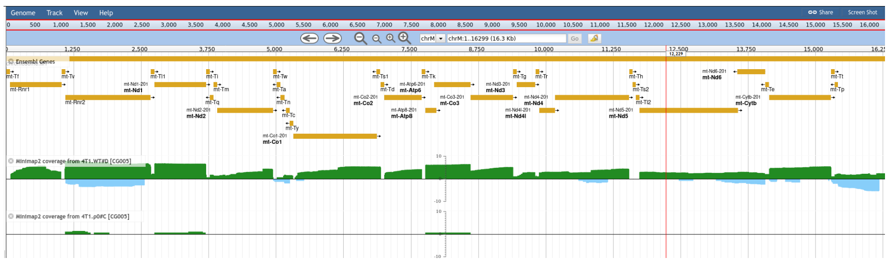
makeMinIONTemplate.sh

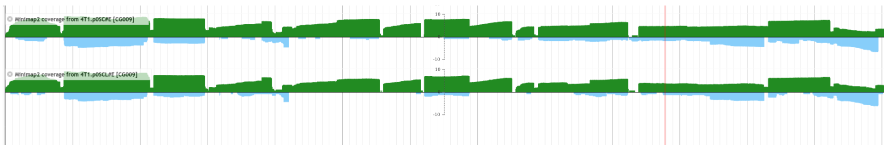
Sanity Check

7 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 the *Expected Results* shown here.

If not, check for the following issues:

- Tracks not displaying at all in JBrowse -- make sure track IDs inside square brackets are of the form [*tracks.<unique-id-without-dots>-track*]
- JBrowse track is reflected in the X axis -- make sure that the reverse bedgraph file is orientated the correct way; it should be created with the '-f 0x10' flag (no capitalisation).
- JBrowse track only shows one direction -- make sure that the reverse bedgraph file has *negative* values, and re-generate the bigwig file


Genome
Track
View
Help
Share
Screen Shot
Ensembl Genes
Minimap2 coverage from 4T1.WT#D [CG005]
Minimap2 coverage from 4T1.p0#C [CG005]


Minimap2 coverage from 4T1.pDSC#E [CG009]
Minimap2 coverage from 4T1.pDSCL#E [CG009]

Stranded BigWig JBrowse tracks for four samples, demonstrating log expression on the mitochondrial chromosome.