Low-frequency variant calling from high-quality mtDNA sequencing data

Marita A. Isokallio,James Stewart

Abstract

High-quality, highly enriched mitochondrial DNA (mtDNA) sample enables detection of extremely low-frequency mtDNA variants. When mtDNA is carefully enriched, fragmented in the presence of EDTA and sequenced with unique dual indices by Illumina HiSeq in paired-end mode, it is possible to reliably detect variants at allele frequencies <0.05 %. To be able to detect variants at every position of the mtDNA genome, 'dual' alignment and variant calling strategy is required.

Citation: Marita A. Isokallio,James Stewart Low-frequency variant calling from high-quality mtDNA sequencing data. **protocols.io**
dx.doi.org/10.17504/protocols.io.nfkdbkw
Published: 19 Apr 2018

Guidelines

The unique dual indices require paired-end sequencing and produce R1 and R2 reads. However, to our experience, R2 reads seem to cause more artifactual variant results (mainly GC>TA indicative of oxidative damage). Thus, only use of R1 reads for low-frequency variant detection is recommended.

If 'Before starting' sample quality requirements cannot be met, more stringent variant calling thresholds - namely allele frequency threshold - may be required and extremely low-frequency variant calling is not possible due to the chemical/biological artifacts present in the sample. Furthermore, it is always advisable to use control samples and confirm the accuracy, preferably in multiple experiments (meaning re-preparation of the libraries from the same case and control samples and sequencing the samples in another sequencing run).

Before start

Ensure that your mtDNA sample used for the sequencing

1) is of high quality and highly enriched from nuclear DNA (nDNA) contamination,

2) 1 mM of EDTA is included into the sonication step of the Illumina library preparation,

3) unique dual indexing is used and reads are sequenced in paired-end mode, and

4) reads are de-multiplexed based on both indices.

Addition of EDTA (or repair enzymes) has been recommended in order to diminish potential oxidative damage or propagation of other damages during library PCR (Costello et al. 2012, Chen et al. 2017). Whereas dual-indexing significantly reduces the possibility of between-sample cross-contamination (Kircher et al. 2012).

References

Chen, L. et al., 2017. DNA damage is a pervasive cause of sequencing errors, directly confounding variant identification. *Science*, 355, pp.752–756.

Costello, M. et al., 2013. Discovery and characterization of artifactual mutations in deep coverage targeted capture sequencing data due to oxidative DNA damage during sample preparation. *Nucleic Acids Res.*, 41(6), pp.1–12.

Kircher, M., Sawyer, S. & Meyer, M., 2012. Double indexing overcomes inaccuracies in multiplex sequencing on the Illumina platform. *Nucleic Acids Res.*, 40(1), p.e3.

Protocol

Read trimming

Step 1.

R1 reads are trimmed for minimum length, minimun base quality and for TruSeq adapter sequences.

SOFTWARE PACKAGE (Unix)

Flexbar, 2.5

Dodt, M., Roehr, J.T., Ahmed, R., Dietrich, C.
https://sourceforge.net/projects/flexbar/files/

cmd COMMAND (Unix)

```
flexbar -n 16 -r /path/to/reads1.fastq.gz -t output_prefix -f i1.8 -j -z GZ -q 28 -m 50 -a /path/to/adapters.txt -ao 10 -at 1 -ae ANY > file.log.txt
```

-n number of cores -r path to R1 reads -t prefix for output -f quality format -j produce read length file -z GZ compressed output -q minimum base quality for 3' end -m minimun read length (bp) -a adapter file -ao minimum adapter sequence overlap -at allowed mismatches in the adapter sequence per 10 bp -ae adapter trimming end

Read alignment to NORMAL reference genome

Step 2.

Reads are first aligned to normal mouse mtDNA reference genome.

SOFTWARE PACKAGE (Unix)

SAMtools, 1.3.1

Li, H. et al.
http://samtools.sourceforge.net/

cmd COMMAND (normal reference genome)

```
# Create index for the reference genome (required only once)
bwa index -p reference reference.fa

# bwa mem alignment
bwa mem -t 15 -P -T 19 -B 3 -L 5,4 /path/to/reference/reference
/path/to/reads1_flexbar.fastq.gz > reads1.sam

# Sorting and indexing
samtools view -Sbu reads1.sam | samtools sort - -T reads1.sorted -o
reads1.sorted.bam

samtools index reads1.sorted.bam
```

-t threads -P single-end reads -T minimum score to output -B allowed mismatches -L penalty for 5' and 3' end clipping

NOTES

Marita A. Isokallio 25 Feb 2018

This will allow variant detection on positions 200 to 16099 i.e. on the full genome excluding the circular genome junction region.

Read alignment to a SPLIT reference genome

Step 3.

Generate a split reference genome.

NOTE: this step only needs to be run once and the generated files can be used for the next analysis.

cmd COMMAND (split reference genome)

```
## Create the split reference genome (required only once) ##
# Create variables
split_genome=reference_split.fa
single_line=reference_singleline.fa
split_data=reference_split.data

# Transform the reference fasta file to contain the sequence as a single line
header=$(cat /path/to/reference.fa | grep '>')
cat /path/to/reference.fa | grep -v '>' | tr -d '\n' > $single_line

# Calculate the length of the input reference genome sequence to determine the cutting posi
tion
half_split=$(cat $single_line | awk 'BEGIN {junc=0} junc=int(length($0)/2)
{print junc}')
half_split1=$(echo $half_split | awk '{print $0+1}')
full_len=$(cat $single_line | awk 'BEGIN {len=0} len=length($0) {print len}')
```

```
paste <(echo $half_split) <(echo $half_split1) <(echo $full_len) >
$split_data

# Take the genome halves according to the calculated positions
gen_start=$(cat $single_line | cut -c1-$half_split)
gen_end=$(cat $single_line | cut -c${half_split1}-$full_len)

# Combine the halves in correct order and restore the fasta format
gen_split=$(paste <(echo $gen_end) <(echo $gen_start) | tr -d '\t')
paste <(echo $header) <(echo $gen_split) | tr '\t' '\n' > $split_genome
```

NOTES

Marita A. Isokallio 25 Feb 2018

This will allow variant detection on positions at the circular genome junction region i.e. positions 1-199 and 16100-16299.

Read alignment to a SPLIT reference genome

Step 4.

Similar to **Step 2**, reads are now aligned to the split mouse mtDNA reference genome generated in **Step 3**.

cmd COMMAND (split reference genome)

```
# Create index for the reference genome (required only once)
bwa index -p reference_split reference_split.fa

# bwa mem alignment
bwa mem -t 15 -P -T 19 -B 3 -L 5,4 /path/to/reference/reference_split
/path/to/reads1_flexbar.fastq.gz > reads1_junction.sam

# Sorting and indexing
samtools view -Sbu reads1_junction.sam | samtools sort - -T
reads1_junction.sorted -o reads1_junction.sorted.bam

samtools index reads1_junction.sorted.bam
```

-t threads -P single-end reads -T minimum score to output -B allowed mismatches -L penalty for 5' and 3' end clipping

Read filtering

Step 5.

In order to exclude poorly aligned reads, the aligned reads are filtered for minimum mapping quality.

cmd COMMAND (Unix)

```
# Quality filter only uniquely aligned reads for further processing (bwa mem mapping qualit
y 0 indicates multi-mapping reads)

# First the reads aligned to normal reference genome
samtools view -bq 1 reads1.sorted.bam > reads1.accepted.bam
samtools index reads1.accepted.bam

# Similar to reads aligned to the split reference genome
samtools view -bq 1 reads1_junction.sorted.bam > reads1_junction.accepted.bam

samtools index reads1_junction.accepted.bam
```

-b output BAM format -q minimum mapping quality to keep

NOTES

Marita A. Isokallio 25 Feb 2018

NOTE: Different alignment tools use different mapping qualities, see for example a blog post by Keith Bradnam.

Calculate coverage

Step 6.

Calculate coverage for both alignment files, the one from alignment to the normal reference genome and the one from alignment to the split reference genome.

Subset the results such that the entire genome is represented with correct coverage i.e. use the alignment to the normal reference genome for the positions 200-16099 (mouse genome positions) and the alignment to split reference genome for the junction region positions between -200 and +200.

SOFTWARE PACKAGE (Unix)

BEDTools, 2.22.1

Quinlan, A.R., Hall, I.M.
https://github.com/arq5x/bedtools2

cmd COMMAND

```
# Firts calculate coverage for the normal reference alignment
bedtools genomecov -d -ibam reads1.accepted.bam -g reference.fa > coverage.txt

# Then the same for the split reference alignment
bedtools genomecov -d -ibam reads1_junction.accepted.bam -
g reference_split.fa > coverage_junction.txt

## Combine the coverages in order to represent the entire mtDNA genome

# Take the middle and end points of the split junction genome for extracting correct lines
mid_point=$(cat reference_split.data | awk '{print $2}')
end_point=$(cat reference_split.data | awk '{print $3}')

# Middle part of the normal coverage file
cat coverage.txt | awk -v endpoint="$end_point" \
'$2 > 200 && $2 < endpoint - 200 {print}' > coverage.middle

# Re-coordinate the junction region and take only -200 and +200
cat coverage_junction.txt | awk -v midpoint="$mid_point" \
-v endpoint="$end_point" \
'BEGIN {OFS = "\t"; pos = 0; test = 0; res = 0} \
{pos = $2; test = pos - midpoint; \
if(test <= 0) res = endpoint + test; \
else res = test; $2 = res; print}' | \
sort -nk2 | awk -v endpoint="$end_point" \
'$2 <=200 || $2 >= endpoint - 200 {print}' > coverage.junction_replacement

# Merge the middle and junction regions into a final result file
cat $normal_middle $junction_replacement | sort -nk2 > coverage_final.txt
```

NOTE: awk command is split over multiple rows for better readability. If the command does not run, first try to run it as a single line.

Variant calling

Step 7.

Variant calling is done for both alignment files, the one that was aligned to the normal reference genome and the one that was aligned to the split reference genome.

Variant calling sets minimum base-calling qualities to be considered for the reference and alternative bases and calls indels simultaneosly.

Variants are filtered only for strand bias and minimum quality.

NOTE: Disabling the default run of lofreq filter allows detection of extremely rare variants from a high-quality, highly enriched mtDNA sample.

SOFTWARE PACKAGE (Unix)

SAMtools, 1.3.1

Li, H. et al.

http://samtools.sourceforge.net/

cmd COMMAND (variant calling for normal and split alignments)

```
# First set the indel qualities for using --call-indels
lofreq indelqual --dindel --ref reference.fa --out reads1.indelqual.bam reads1.accepted.bam

lofreq indelqual --dindel --ref reference_split.fa --
out reads1_junction.indelqual.bam  reads1_junction.accepted.bam

# Generate the index for both files
samtools index reads1.indelqual.bam
samtools index reads1_junction.indelqual.bam


# Variant calling including indels
lofreq call-parallel --pp-threads 20 -f reference.fa -o reads1.nofilter.vcf -N -B -q 30 -
Q 30 --call-indels --no-default-filter reads1.indelqual.bam

lofreq call-parallel --pp-threads 20 -f reference_split.fa -
o reads1_junction.nofilter.vcf -N -B -q 30 -Q 30 --call-indels --no-default-
filter reads1_junction.indelqual.bam


# Filtering the results if >85% of variant reads are on single strand and for minimum quali
ty
lofreq filter --no-defaults --snvqual-thresh 70 --indelqual-thresh 70 --sb-incl-indels -
B 60 -i reads1.nofilter.vcf -o reads1.sbfiltered.vcf

lofreq filter --no-defaults --snvqual-thresh 70 --indelqual-thresh 70 --sb-incl-indels -
B 60 -i reads1_junction.nofilter.vcf -o reads1_junction.sbfiltered.vcf
```

--dindel Add Dindel's indel qualities (Illumina specific) --pp-threads number of threads -f reference genome .fasta file -o output file -N Don't merge mapping quality in LoFreq's model -B Disable use of base-alignment quality (BAQ) -q minimum REF base quality -Q minimum ALT base quality --call-indels enable indel calls --no-default-filter Don't run default 'lofreq filter' automatically after calling variants

Variant calling

Step 8.

Combine the called variants from the both VCF files (normal and split reference genomes) and re-coordinate the positions.

First, use the *reference_split.data* file generated in the **Step 3** that describes the middle and last genome position numbers.

From the VCF file that used the split reference genome, take only the variants identified on the genome junction region i.e. positions between -200 and +200. Replace the position numbers with those corresponding the normal reference genome in order to combine these variants with the ones in the VCF file that used the normal reference genome.

From the VCF file that used the normal reference genome, take all variants except those on the above described genome junction region.

Combine the variant lists into a single VCF file that now has the positions according to the normal reference genome and variants are correctly detected also around the junction region.

Finally, separate SNVs and indels into separate files for downstream processing.

SOFTWARE PACKAGE (Unix)

LoFreq, 2.1.2

Wilm A. et al.
https://github.com/CSB5/lofreq

cmd COMMAND (combine variant files)

```
## Combine the original and junction region results for the junction region ##
# Read in the mid and end points of the split genome junction region (generated in Step 3 f
or the split reference genome)
mid_point=$(cat reference_split.data | awk '{print $2}')
end_point=$(cat reference_split.data | awk '{print $3}')

# Re-coordinate the variants and take only -200 and +200 region
cat reads1_junction.sbfiltered.vcf | awk '/#CHROM/ {flag = 1; next} flag {print}' | \
awk -v midpoint="$mid_point" -v endpoint="$end_point" \
'BEGIN {OFS = "\t"; pos = 0; test = 0; res = 0} \
{pos = $2; test = pos - midpoint; if(test <= 0) res = endpoint + test; \
else res = test; $2 = res; print}' | \
sort -nk2 | \
awk -v endpoint="$end_point" '$2 <= 200 || $2 >= endpoint -
 200 {print}' > junction_replacement.vars

# Create intermediate VCF file header
cat reads1.sbfiltered.vcf | \
```

```
awk '/#/ {print}' > reads1.sbfiltered_junction_combined.vcf

# Take middle part of the original alignment variant calls
cat reads1.sbfiltered.vcf | \
awk '/#CHROM/ {flag = 1; next} flag {print}' | \
awk -v endpoint="$end_point" '$2 > 200 && $2 < endpoint - 200 {print}' > reads1.middle.vars

# Combine junction replacement and original middle, sort and append to the original VCF hea
der
cat reads1.middle.vars junction_replacement.vars | \
sort -nk2 >> reads1.sbfiltered_junction_combined.vcf

# Final junction fixed file for SNV and indel separation
# Separate SNVs only and indels only to different files for downstream analysis
lofreq filter --no-defaults --only-snvs \
-i reads1.sbfiltered_junction_combined.vcf -o reads1.snvs.vcf
lofreq filter --no-defaults --only-indels \
-i reads1.sbfiltered_junction_combined.vcf -o reads1.indels.vcf
```

NOTE: awk command is split over multiple rows for better readability. If the command does not run, first try to run it as a single line.

Variant filtering

Step 9.

Variants can be further filtered and annotated with snpEff.

First, make sure that you have the snpEff config file correctly set up to use the mitochondrial codon table (use the same genome version as you used for the alignment).

Then, filter the variants with wanted thresholds. Here, we use the minimum number of total alternative reads (calculated as depth * allele frequency) together with minimum number of alternative reads on each strand (based on DP4 values).

In the next step, we apply an additional filter for variants with extreme strand-bias Phred score (SB>1000), which were not filtered out in the earlier step.

Simultaneously, we filter the results for the near-homoplasmic variants that has expanded in our inbred mouse strain (this part should be omitted if your mouse strain does not carry these variants).

SOFTWARE PACKAGE (Unix)

SnpEff, 4.2

Cingolani P. et al.
https://github.com/pcingola/SnpEff

cmd COMMAND (variant filtering)

```
# Filter variants for minimum number of supporting reads in total and on both strands
cat reads1.snvs.vcf | \
java -jar /software/snpEff/4.2/SnpSift.jar filter \
"( (DP*AF >= 15) & (DP4[2] >= 3) & (DP4[3] >= 3) )" \
```

```
> reads1.filtered.vcf

# Filter variants for mouse strain specific and highly strand biased variants
cat reads1.filtered.vcf | \
java -jar /software/snpEff/4.2/SnpSift.jar filter \
"( (SB < 1000) & (POS != 4891) & (POS != 9461) & (POS != 9027) )" \
> reads1.pos_filtered.vcf
```