HOW TO ALIGN DOVETAIL™ OMNI-C™ READS AND CALL SNPs

Introduction

The purpose of this document is to provide a step-by-step guide for aligning proximity ligation data generated using the Dovetail™ Omni-C™ Kit. It also outlines the process for SNP calling using Omni-C™ data.

Omni-C Data Alignment

The first step is to index the reference assembly with BWA. The only input file required for this step is the reference assembly FASTA file. In this example, we are using "Chr20.p_ctg.fasta" file located in the directory named assembly.

```
bwa index assembly/chr20.p_ctg.fasta
```

After indexing the sequence file, you can move forward with the alignment and QC of the Omni-C data using the omni-c_qc script with utilizes BWA-MEM as follows:

```
~/scripts/omni-c_qc.bash assembly/chr20.p_ctg.fasta ~/data/omnic_reads/chr20_R1.fastq.gz
~/data/omnic_reads/chr20_R2.fastq.gz alignment.bam NA12878
```

In the above command:

- Input files:
  - Sequence file: chr20.p_ctg.fasta file located in the assembly folder
  - Omni-C data: chr20_R1.fastq.gz and chr20_R2.fastq.gz located in the omnic_reads folder
- Output file: alignment.bam NA12878
  - NA12878 represents the sample name and can be customized – It is required to have a sample name for the bam output file. This is an identifier that some tools such as GATK use
- omni-c_qc.bash script can be found here: https://github.com/dovetail-genomics/omni-c_qc
- Parameters: in the alignment pipeline, we use -5SP -T0 parameters for BWA-MEM. These parameters offer the following functions:
  - -5: marks the first coordinate at 5-prime end as primary alignment
  - -S: skips rescuing mates that are not mapped in insert size limit
  - -P: skips read pairing

- -T: represents the minimum mapq to output an alignment, here set to 0
- SAMBLASTER is also run as part of the omni-c_qc.bash script to mark PCR dups in the resulting alignments

Calling SNPs/Indels With Omni-C Data

Since Omni-C data exhibit a similar coverage profile as standard Illumina paired-end libraries, they can be used for SNP calling. There are numerous tools available for SNP calling with Illumina data. Here, we are using GATK. To run GATK, you first need to create a dictionary (*.dict) file for your sequence FASTA file as follows:

```
picard CreateSequenceDictionary R=assembly/chr20.p_ctg.fasta O=assembly/chr20.p_ctg.dict
```

Then, run GATK HaplotypeCaller as follows:

```
~/software/gatk-4.1.5.0/gatk HaplotypeCaller -R assembly/chr20.p_ctg.fasta
-I alignment.bam -O raw.g.vcf.gz  -ERC GVCF  --native-pair-hmm-threads 1 --max-alternate-
alleles 3  -contamination 0  -L   scaffold_1:1-350000
```

In the above command:

- -R: specifies the sequence file in FASTA format
- - I: specifies the bam file containing the Omni-C alignments
- -O: specifies the output VCF file to which variants should be written
- -ERC GVCF: provides the confidence score for each variant in the VCF file
- --native-pair-hmm-threads: specifies the number of threads, here set to 1
  - Increasing the thread number can reduce the total processing time
- --max-alternate-alleles: specifies the maximum number of alternate alleles to genotype. It should be set to 3 to call SNPs and short indels only. Increasing the number of alternate alleles to genotype will call SNPs, short and large indels
- -contamination: is set to 0 as the sequencing data contain no contaminant sequences, such as microbial or host sequences
- -L: used if you want to call variants in a specific genomic region, here set as scaffold_1:1-350000. If you omit this parameter, the variants will be called on all the sequences in the FASTA file

Please note, HaplotypeCaller works well for the human genome. If you are working with non-human samples, FreeBayes is a better alternative. It is much faster than HaplotypeCaller and can be further parallelized. Here are the steps if you wish to use FreeBayes in parallel:

First, index the sequence file

```
samtools faidx assembly/chr20.p_ctg.fasta
```

This returns an index file ending in "*.fai". Then, run the FreeBayes tool in parallel as follows:

```
freebayes-parallel <(fasta_generate_regions.py assembly/chr20.p_ctg.fasta.fai 100000) 40 -f
assembly/chr20.p_ctg.fasta alignment.bam  > freebayes_variants.vcf
```

In the above command:

- Input files:
    - chr20.p_ctg.fasta.fai
    - alignment.bam
    - chr20.p_ctg.fasta
- Output file: freebayes_variants.vcf
- Window size (bin): set to 100,000 bp

This should take around 15 minutes on a 40-core machine. If you want to run on a specific region, the command would be:

```
freebayes -f assembly/chr20.p_ctg.fasta  -r scaffold_1:1-350000 alignment.bam  > small.vcf
```

- Input files:
    - chr20.p_ctg.fasta
    - alignment.bam
- Output file: small.vcf
- -r: specifies the region of interest, here set as scaffold_1:1-350000