Structural Variant Analysis

Leila Taher

WS 2022/2023

Contents

1 Introduction

The aim of this tutorial is to call structural variants (SVs) in a human genome by identifying discordant paired-end read mappings and split reads.

Discordant paired-end alignments conflict with the alignment patterns that we expect (i.e., concordant alignments) for the DNA library and sequencing technology we have used.

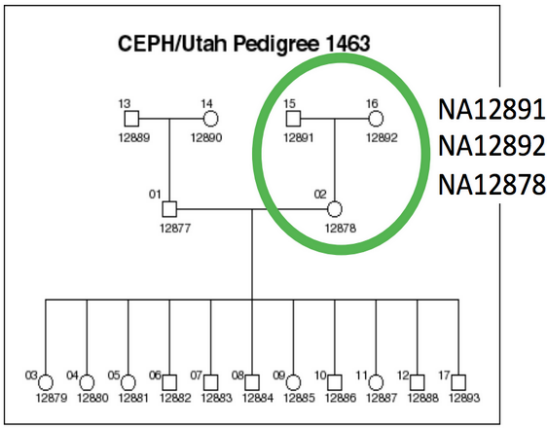
For example, given a ~500bp paired-end Illumina library, we expect pairs to align in F/R orientation and we expect the ends of the pair to align roughly 500bp apart. Pairs that align too far apart suggest a potential deletion in the DNA sample's genome. As you may have guessed, the trick is how we define what "too far" is — this depends on the fragment size distribution of the data.

Split-read alignments contain SV breakpoints and consequently, then DNA sequences up- and down-stream of the breakpoint align to disjoint locations in the reference genome.

We will be working with data from The 1000 Genomes Project. In particular, we will be using samples NA12878, NA12891, and NA12892.

- You can find information on the samples in the data portal of The 1000 Genomes Project

These samples are part of the Illumina Platinum Genomes Project, which is a 50X-coverage whole-genome HiSeq Illumina sequencing dataset. The original data can be downloaded from the following URL.



For practical reasons, we subsampled the reads from the sample because running the whole dataset would take way too much time and resources. We are going to focus on the reads extracted from the chromosome 20.

1.1 Tools

- samtools
  `sudo apt-get install samtools`

- Picard
  `sudo apt-get install picard-tools`

- delly
  `sudo apt-get install delly`

- bcftools
  `sudo apt-get install bcftools`

- tabix
  `sudo apt install tabix`

- IGV
  `sudo apt-get install igv`

DELLY is a tool that can discover, genotype and visualize deletions, tandem duplications, inversions and translocations at single-nucleotide resolution in short-read massively parallel sequencing data. It uses paired-end and split-reads. If you are interested in how the method works, you can read the original publication here. Install the tool if necessary.

1.2 Data files

Sample	BAM file
NA12878	NA12878_chr20.bam
NA12891	NA12891_chr20.bam
NA12892	NA12892_chr20.bam

2 Mapping with BWA-MEM

In the interest of time, the reads were already mapped for you. BWA is a software package for mapping low-divergent sequences against a reference genome. It consists of three algorithms: BWA-backtrack, BWA-SW and BWA-MEM. The first algorithm is designed for Illumina sequence reads up to 100bp, while the rest two for longer sequences ranged from 70bp to 1Mbp. BWA-MEM and BWA-SW share similar features such as long-read support and split alignment, but BWA-MEM, which is the latest, is generally recommended for high-quality queries as it is faster and more accurate. BWA-MEM reports both paired-end and split-read alignments in a single BAM output file, which is convenient for the analysis.

In the alignment commands, note that the use of the `-M` parameter is used to mark shorter split hits as secondary. If split short reads are not marked as secondary hit some tools may treat them as a primary read and this could lead to errors or biases in some of the metrics.

Reads were mapped to the soft-masked GRCh38 assembly of the human genome (Ensembl release 102 ftp://ftp.ensembl.org/pub/release-102/fasta/homo_sapiens/dna/).

```
bwa mem <REF_ASSEMBLY.fa> \
Reads_Forward.fq \
Reads_Reverse.fq \
-M -t 2 | samtools view -S -b -
```

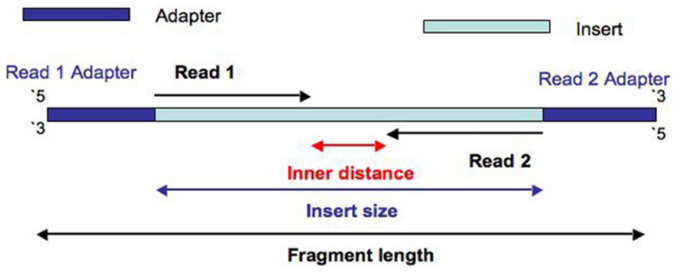
1. Download the `bam` files from the teach center
2. Download the reference assembly
    1) connect to the ensembl server: `ftp ftp.ensemble.org`
    2) navigate to the correct directory with the `cd` command
    3) download the assembly with the `get` command
3. Index the reference sequence

    ```
    samtools faidx <REF_ASSEMBLY.fa>
    ```

4. Use `samtools` to sort and index the `bam` files.

3 Characterizing the fragment size distribution

In order to identify discordant mappings of paired-end reads, we must first characterize the insert size distribution with `picard` **(A)** and with `samtools` **(B)**.



We will decide on the threshold for discordant mappings based on the distribution of insert sizes.

5. **(A)** Using `picard`: The following commands extract pairs of forward and reverse reads from a BAM file and compute the mean and standard deviation of the insert size.

```
PicardCommandLine CollectInsertSizeMetrics \
I=<input.bam> O=<insert_size_metrics.txt> \
H=<insert_size_histogram.pdf>
```

   a. Use R to visualize the insert size distribution. Repeat for all remaining samples.
   b. What is the mean and standard deviation of the insert size? Is it the same for all samples?

6. **(B)** Using `samtools`: The ninth column of a SAM file, observed Template LENgth (TLEN), can be used as an approximate of the fragment length. Let us obtain the insert sizes using only the first pair of *properly mapped pairs* (flag `-f66`, see Decoding SAM flags).

   ```
   samtools view -f66 input.bam | cut -f 9 > insert-sizes.txt
   ```

   a. Why are there negative values?
   b. Use R to visualize the distribution of insert sizes and get some summary statistics.

7. The insert size estimation using this method has a limitation: it merely reflects the distance between the mappings. In particular, the method may provide misleading estimates for RNA-seq data (Why?).

8. Exclude very large, unlikely inserts and recalculate mean and standard deviation.

For each input BAM file, DELLY computes the default read-pair orientation and the paired-end insert size distribution characterized by the median and standard deviation of the library. Based on these parameters, DELLY then identifies all discordantly mapped read-pairs that either have an abnormal orientation or an insert size greater than the expected range. DELLY hereby focuses on uniquely mapping paired-ends and the default insert size cutoff is three standard deviations from the median insert size (*Rausch et. al*).

9. Based on the file generated for sample NA12891 in 5, what would you say DELLY will use as a threshold for discordant mappings?

   - hint: you will need to pipe the file into `awk` to print the column number corresponding to the MEDIAN_INSERT_SIZE and STANDARD_DEVIATION

4 SV detection

Now we will call germline SVs on each sample. The steps are adapted from the DELLY README file.

10. The notation of the chromosome will cause an error. Change the first line in the reference file `Homo_sapiens.GRCh38.dna_sm.chromosome.20.fa` from `>20` to `>chr20` using unix's `sed` command.

11. For each sample, type:

```
delly call -g <REF_ASSEMBLY.fa> -o <FN.bcf> \
        -x human.hg38.excl.tsv \
        <INPUT.bam>
```

   - If you omit the reference sequence, DELLY skips the split-read analysis. To save runtime it is advisable to exclude telomere and centromere regions. For human, DELLY ships with such an exclude list. For the human genome, the file is called `human.hg38.excl.tsv` and can be downloaded from the teachcenter.
   - `.bcf` files are compressed `.vcf` files. DELLY's output files follow the vcf specification. All fields in the file are explained in the header.
   - To look at the output you can use bcftools:

```
bcftools view output.bcf | less -S
```

12. generate a file with

```
touch delly_metrics.txt
```

Pipe the following outputs of each `bcf` file into the file. Don't forget to use » to append.

a.) the file name
b.) the number of SVs did you find in each sample
c.) the number of each type of SV (e.g., deletions, duplications, inversions) in each sample

4.1 Merging SV calls

13. Merge the SVs from multiple samples into a single, unified site list with DELLY:

```
delly merge -m 500 -n 1000000 \
-o SVs_merged.bcf \
-b 500 -r 0.5 <FN1.bcf> <FN2.bcf> <FN3.bcf>
```

`-m` and `-n` are the minimum and maximum SV sizes. `-b` is the maximum breakpoint offset. `-r` is the minimum reciprocal overlap. Replace `FN%d.bcf` with the files we generated in 11.

14. Examine the output with bcftools. Pipe the following outputs of the merged `bcf` file into `delly_metrics.txt`.

a.) the file name
b.) the number of SVs
c.) the number of each type of SV (e.g., deletions, duplications, inversions)

4.2 Re-genotyping

15. Genotype this merged SV site list across all samples. Name the output `.bcf` files `NA12878_geno.bcf`, `NA12891_geno.bcf`, and `NA12892_geno.bcf`.

```
delly call -g Reference_Assembly.fa \
-v SVs_merged.bcf \
-o <output.bcf> \
-x human.hg38.excl.tsv <input.bam>
```

16. Examine the output files with `bcftools`. You should see information on the genotype for each individual.

4.3 Merging the re-genotyped calls

17. Merge all re-genotyped samples to get a single .bcf output file using `bcftools merge`:

```
bcftools merge -O b \
-o output.bcf \
NA12878_geno.bcf NA12891_geno.bcf NA12892_geno.bcf
```

- `O b` indicates that the output should be a .bcf file.

18. Index the resulting .bcf file and create a .vcf file for visualization:

```
bcftools index output.bcf
bcftools view output.bcf > output.vcf
bgzip -c output.vcf > output.vcf.gz
tabix -fp vcf output.vcf.gz
```

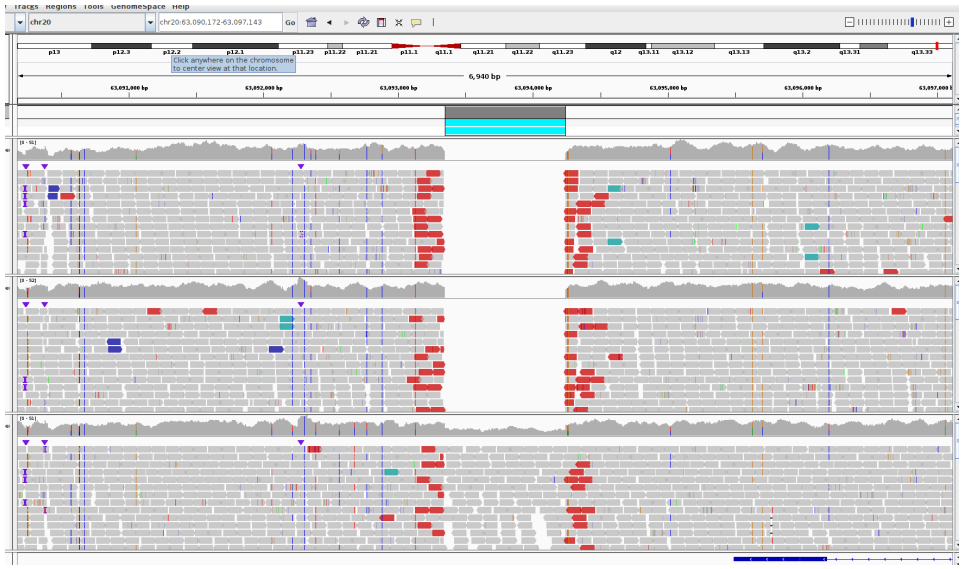
5 Setting up IGV for SV visualization

19. Launch IGV and load the merged SV calls and the mapping files for the individual samples using `File -> Load from File`.

20. Navigate to the following location to see a deletion: `chr20:63,090,172-63,097,143`.

21. Load the .bam files.

    You should see something like this:



    You can try to configure IGV such that we can more clearly see the alignments that support the SV prediction(s).

    - Color the alignments by insert size and pair orientation (right click on reads for coloring options).

5.1 Explore the SVs

22. Is the variant located at chr20:52,142,500-52,145,000 found in any member of the trio? If so which genotype do they have?

23. How would you locate all deletions in which at least one member of the trio is
    a) homozygous for an alternate allele
    b) heterozygous for an alternate allele and
    c) homozygous for the reference allele?

6 Acknowledgements

This tutorial is heavily based on the Structural Variant Calling module created by Mathieu Bourgey, Ph.D, and modified by Pascale Marquis.

7 Versions

```
samtools 1.13
Using htslib 1.13

picard-tools 2.18.25+dfsg-2

Delly version: v0.8.2
using Boost: v1.71.0
using HTSlib: v1.10.2-3ubuntu0.1

bcftools 1.10.2
Using htslib 1.10.2-3ubuntu0.1

tabix (htslib) 1.13
```