# 1 Structural Variation Calling

## 1.1 Introduction

Structural variants (SVs) are large genomic alterations of at least 50 bp or larger in size. There are several types of SVs, including deletions, duplications, insertions, inversions, and translocations which describe different combinations of DNA gains, losses, or rearrangements. Copy number variations (CNVs) are a particular subtype of SV mainly represented by deletions and duplications. SVs are typically described as single events, although more complex scenarios involving combinations of SV types exist.

## 1.2 Learning outcomes

On completion of the tutorial, you can expect to be able to:

- Call structural variants using standard tools
- Visualise structural variants using standard tools
- Call structural variants from long read data
- Use bedtools to do regional comparisons over genomic co-ordinates

## 1.3 Tutorial sections

This tutorial comprises the following sections: 1. Looking at structural variants in VCF 2. Calling structural variants
3. Structural variants from long reads
4. Bedtools

## 1.4 Authors

This tutorial was written by Jacqui Keane based on material from Thomas Keane.

## 1.5 Running the commands in this tutorial

You can follow this tutorial by typing all the commands you see into a terminal window. This is similar to the "Command Prompt" window on MS Windows systems, which allows the user to type DOS commands to manage files.

To get started, open a new terminal window and type the command below:

[1]:
```
cd /home/manager/course_data/structural_variation/data
```

## 1.6 Let's get started!

This tutorial requires that you have breakdancer, lumpy, minimap2, sniffles, bedtools and igv installed on your computer. These are already installed on the virtual machine you are using. To check that these are installed, run the following commands:

[2]:
```
breakdancer-max --help
```

[ ]:
```
lumpy --help
```

[ ]:
```
minimap2
```

[ ]:
```
sniffles
```

[ ]:
```
bedtools
```

This should return the help message for software breakdancer, lumpy, minimap2, sniffles and bedtools respectively.

If after this course you would like to download and install this software the instructions can be found at the links below, alternatively we recommend bioconda for the installation and management of your bioinformatics software.

- The breakdancer website
- The lumpy github page
- The minimap2 website
- The sniffles website
- The bedtools website

To get started with the tutorial, go to the first section: Looking at structural variants in VCF

# 2 Looking at Structural Variants in VCF

Structural variants can be stored in VCF files. In this part of the tutorial, we will look at how these are represented in a VCF file.

First, check you are in the correct directory:

[1]:
```
pwd
```

It should display something like:

```
/home/manager/course_data/structural_variation/data
```

Navigate to the `exercise1` directory:

[ ]:
```
cd exercise1
```

There is a VCF file called `ERR1015121.vcf` that was produced using the Lumpy SV calling software. Look at the VCF file using the `less` command and answer the questions that follow:

[ ]:
```
less ERR1015121.vcf
```

## 2.1 Excercises

1. What does the CIPOS format tag indicate?
2. What does the PE tag indicate?
3. What tag is used to describe an inversion event?
4. What tag is used to describe a duplication event?
5. How many deletions were called in total? (**Hint:** DEL is the info field for a deletion. The -c option of the grep command can be used to return a count of matches.)
6. What type of event is predicted at IV:437148? What is the length of the SV? How many paired-end reads and split-reads support this SV variant call?
7. What is the total number of SV calls predicted on the IV chromosome?

Now continue to the next section of the tutorial: Calling structural variants

# 3 Calling Structural Variants

There are several software tools available for calling structural variants. We will use two callers in this part of the tutorial, `breakdancer` and `lumpy`

## 3.1 Breakdancer

BreakDancer predicts five types of structural variants: insertions (INS), deletions (DEL), inversions (INV), inter-chromosomal translocations (CTX) and intra-chromosomal translocations (ITX) from next-generation short paired-end sequencing reads using read pairs that are mapped with unexpected separation distances or orientation.

Navigate to the `exercise2` directory:

[ ]:
```
cd ../exercise2
```

[ ]:
```
ls
```

We will use the Breakdancer software package to call structural variants on a yeast sample that was paired-end sequenced on the illumina HiSeq 2000. Breakdancer first needs to examine the BAM file to get information on the fragment size distribution for each sequencing library contained in the BAM file.

The breakdancer.config file has information about the sequencing library fragment size distribution. Use the `cat` command to print the contents of the `breakdancer.config` file.

[ ]:
```

```

**Q What is the mean and standard deviation of the fragment size?**

Run the breakdancer SV caller using the command:

[ ]:
```
breakdancer-max breakdancer.config > ERR1015121.breakdancer.out
```

Look at the output of Breakdancer.

[ ]:
```
head ERR1015121.breakdancer.out
```

Note that the output from Breakdancer is NOT VCF format, instead it is a simple text format with one line per SV event.

### 3.1.1 Exercises

1. What type of SV event is predicted at position III:83065?
2. What is the size of this SV?
3. What is the score of this SV?
4. What type of SV event is predicted at position II:258766?
5. Convert the output of breakdancer into BED format

The BED format is explained here: `https://genome.ucsc.edu/FAQ/FAQformat.html#format1`

To complete this task, create a command that: 1. Extracts all the deletions from the breakdancer.out file (**Hint:** use grep) 2. Prints columns: 1, 2, 5, 7, and 9 to create a BED file with columns: chromosome, start, end, name, and score. (**Hint:** use awk to do this, e.g. `awk '{print $1"\t"$2}'`) 3. Print the resulting bed output into a file called: breakdancer.dels.bed

## 3.2 Inspecting SVs with IGV

Now we will open the IGV genome browser and inspect some of the predicted structural variants.

To do this type:

[ ]:
```
igv
```

Open the reference genome. **Go to ' *Genomes -> Load Genome From Server…* ' and select "S. cerevisiae EF3 r62".**

Load the BAM file. **Go to ' *File -> Load from File…* '. Select the "ERR1015121.bam" BAM file and click' *Open* '.**

Load the BED file for the deletion calls that you created in the exercise 5 above. **Go to ' *File -> Load from File…* '. Select the "breakdancer.dels.bed" BED file and click' *Open* '.**

### 3.2.1 Exercises

Using the navigation bar, go to region II:257,766-259,766.

1. Can you see the structural variant? What type of structural variant is it? (**Hint:** you may need to zoom out a little to see the full structural variant).
2. Can you see any evidence to support this SV call?
3. Can you estimate the size of the SV?

The VCF in the `exercise1` directory was produced by another structural variant caller on the same sample as this exercise.

4. Load the `exercise1/ERR1015121.vcf` VCF into IGV also (File - Load from file, and select `ERR1015121.vcf` in the exercise 1 directory).
5. Was the structural variant at II:258766 also called by the other structural variant software (lumpy)?

Using the navigation bar, go to to region II:508,064-511,840.

6. Is there a SV deletion called in this region by either SV caller?
7. Is there any read support for a SV deletion in this region? If so, how many read pairs could support the deletion call (**Hint:** change the IGV view to `squished` and `View as pairs` to see any inconsistently aligned read pairs).

## 3.3 Lumpy

We will use the Lumpy software package to call structural variants on a yeast sample that was paired-end sequenced on the Illumina Hiseq 2000. Lumpy is designed to take BAM files that have been aligned with `BWA-mem`.

Navigate to the `exercise 3` directory:

[ ]:
```
cd ../exercise3
```

[ ]:
```
ls
```

Check that there is a BAM file called `ERR1015069.bam` and an index file `ERR1015069.bam.bai` in the directory. The sequence data has already been mapped with `bwa mem` and the results are stored in `ERR1015069.bam`.

The first step for running Lumpy is to extract the read pairs that are discordantly mapped (i.e. pairs that are not mapped within the expected fragment size distribution). Use Samtools to extract these reads:

[ ]:
```
samtools view -bh -F 1294 ERR1015069.bam | samtools sort -O bam -T ERR1015069.
↪temp -o ERR1015069.discordants.bam
```

Now index the bam file. **Hint:** use samtools index

[ ]:
```

```

### 3.3.1 Exercises

1. What does the `-F` option in `samtools view` do?
2. Which BAM flags does 1294 indicate? (**Hint:** in your web browser, visit https://broadinstitute.github.io/picard/explain-flags.html and enter 1294 to find out)

The next step for Lumpy is to extract the reads that are only split mapped (i.e. split read alignments). This is all one single command:

[ ]:
```
samtools view -h ERR1015069.bam | extractSplitReads_BwaMem -i stdin | samtools␣
↪view -b - | samtools sort -O bam -T ERR1015069.temp -o ERR1015069.splitters.
↪bam
```

Now index the bam file. (**Hint:** use samtools index)

[ ]:
```

```

Finally, call structural variants using lumpy, providing it with the original BAM file and the two BAM files we prepared earlier.

[ ]:
```
lumpyexpress -B ERR1015069.bam -S ERR1015069.splitters.bam -D ERR1015069.
↪discordants.bam -o ERR1015069.vcf
```

### 3.3.2 Exercises

3. What type of SV event occurs at position IV:383993? What is the length of the SV event?
4. What type of SV event occurs at position XV:43018? What is the length of the SV event?

Congratulations, you have sucessfully called structural variants from some NGS data. Now continue to the next section of the tutorial: Calling structural variants from long reads

# 4 Calling Structural Variants from Long Reads

In this part of the tutorial we will use long read data to identify structural variants using the SV caller Sniffles.

First navigate to the `exercise4` directory:

[ ]:
```
cd ../exercise4
```

[ ]:
```
ls
```

## 4.1 Introducing the dataset

We will use data from a Saccharomyces cerevisiae strain (YPS128) that was sequenced at the Wellcome Sanger Institute and deposited in the ENA (Project: PRJEB7245, sample: SAMEA2757770, analysis: ERZ448241).

The sequencing reads are contained in a fastq file:

`YPS128.filtered_subreads.10x.fastq.gz`

The reference genome is in the ref directory in a fasta file:

`Saccharomyces_cerevisiae.R64-1-1.dna.toplevel.fa`

## 4.2 Align the data

Before you can use Sniffles to call SVs, it is very important that the reads are aligned with an aligner suitable for long reads.

The software minimap2 is a long-read aligner designed to align PacBio or Oxford Nanopore (standard and ultra-long) to a reference genome.

You can find the usage of minimap2 by typing:

[ ]:
```
minimap2
```

Align the reads with minimap2 and send the output to a SAM file called YPS128.10x.filtered_subreads.sam.

[ ]:
```

```

**Note:** use the -t parameter to use multiple threads in parallel (this will increase the speed of the alignment by using more than one CPU core, I suggest using 2).

Convert the output to BAM format **Hint:** use samtools view -b

[ ]:
```

```

Sort the BAM file and produce a sorted BAM file called `YPS128.10x.filtered_subreads.sorted.bam`. **Hint:** Use samtools sort.

[ ]:
```

```

Use samtools calmd to calculates MD and NM tags. This enables variant calling without requiring access to the entire original reference.

[ ]:
```
samtools calmd YPS128.filtered_subreads.10x.fastq.sorted.bam ../ref/
↪Saccharomyces_cerevisiae.R64-1-1.dna.toplevel.fa |samtools view -o YPS128.
↪filtered_subreads.10x.fastq.sorted.calmd.bam -O bam -
```

Finally, use samtools to index this BAM file. **Hint** Use samtools index

[ ]:
```

```

## 4.3 Call structural variants

Sniffles is a structural variation (SV) caller that is designed for long reads (Pacbio or Oxford Nanopore). It detects all types of SVs (10bp+) using evidence from split-read alignments, high-mismatch regions, and coverage analysis. Sniffles takes the BAM file as input and outputs VCF.

To find the usage for Sniffles, type:

sniffles

Using the default parameters, call SVs with Sniffles and output the results to a VCF file called `YPS128.10x.vcf`.

[ ]:
```

```

**Hint:** You don't need to change any of the default parameters, but you will need to work out how to provide the input BAM file and specify the output VCF file. The documentation on sniffles is here : `https://github.com/fritzsedlazeck/Sniffles/wiki/Parameter`.

## 4.4 Inspecting SVs with IGV

Open IGV:

[ ]:
```
igv
```

Open the reference genome `Saccharomyces_cerevisiae.R64-1-1.dna.toplevel.fa` and load the VCF file `YPS128.10x.vcf`. Now answer the questions that follow.

## 4.5 Exercises

1. What sort of SV was called at on chromosome 'Mito' at position 29295?
2. What is the length of the SV?
3. How many reads are supporting the SV?
4. From a visual inspection of the SV in IGV, can you determine how accurate is the breakpoint of the called SV compared to what you see in IGV? [**Hint:** Better directions on what to do here.]

Now continue to the next section of the tutorial: bedtools

# 5 Bedtools

Bedtools is an extremely useful tool for doing regional comparisons over genomic co-ordinates. It has many commands for doing region based comparisons with BAM, VCF, GFF, BED file formats.

To see the list of commands available, on the command line type:

[ ]:
```
bedtools
```

Navigate to the `exercise5` directory.

[ ]:
```
cd ../exercise5
```

[ ]:
```
ls
```

In this directory, there are two VCF files and the yeast genome annotation in GFF3 format `Saccharomyces_cerevisiae.R64-1-1.82.genes.gff3`.

## 5.1 bedtools intersect

Given two sets of genomic features, the `bedtools intersect` command can be used to determine whether or not any of the features in the two sets "overlap" with one another. For the intersect command, the -a and -b parameters are used to denote the input files A and B.

For example, to find out the overlap between the SVs in `ERR1015069.dels.vcf` and the annotated region of the yeast genome try

[ ]:
```
bedtools intersect -a ERR1015069.dels.vcf -b Saccharomyces_cerevisiae.R64-1-1.
↪82.genes.gff3
```

This command reports the variant in the file `ERR1015069.dels.vcf` every time it overlaps with a feature in `Saccharomyces_cerevisiae.R64-1-1.82.genes.gff3`. Therefore if a variant overlaps more than one feature it will be reported more than once. To report the unique set of variants use:

[ ]:
```
bedtools intersect -u -a ERR1015069.dels.vcf -b Saccharomyces_cerevisiae.R64-1-
1.82.genes.gff3
```

The default is to report overlaps between features in A and B so long as at least one base pair of overlap exists. However, the `-f` option allows you to specify what fraction of each feature in A should be overlapped by a feature in B before it is reported.

To specify a more strict intersect and require at least 25% of overlap of the SV with the genes use the command:

[ ]:
```
bedtools intersect -u -f 0.25 -a ERR1015069.dels.vcf -b␣
↪Saccharomyces_cerevisiae.R64-1-1.82.genes.gff3
```

The `bedtools intersect` command can also be used to determine how many SVs overlap between two VCF files. For more information about `bedtools intersect` see the help:

[ ]:
```
bedtools intersect -h
```

## 5.2 bedtools closest

Similar to intersect, `bedtools closest` searches for overlapping features in A and B. In the event that no feature in B overlaps the current feature in A, closest will report the nearest (that is, least genomic distance from the start or end of A) feature in B.

An example of the usage of `bedtools closest` is:

[ ]:
```
bedtools closest -d -a ERR1015069.dels.vcf -b Saccharomyces_cerevisiae.R64-1-1.
↪82.genes.gff3
```

This command will list all the features in the file `Saccharomyces_cerevisiae.R64-1-1.82.genes.gff3` that are closest to each of the variants in `ERR1015069.dels.vcf`.

The `-d` option means that in addition to the closest feature in `Saccharomyces_cerevisiae.R64-1-1.82.genes.gff3`, the distance to the variant in `ERR1015069.dels.vcf` will be reported as an extra column. The reported distance for any overlapping features will be 0.

For example, to find the closest gene to the variant found at position 43018 on chromosome XV, try

[ ]:
```
bedtools closest -d -a ERR1015069.dels.vcf -b Saccharomyces_cerevisiae.R64-1-1.
↪82.genes.gff3| grep XV | grep 43018
```

For more information about `bedtools closest` see the help:

[ ]:
```
bedtools closest -h
```

## 5.3 Exercises

1. How many SVs found in `ERR1015069.dels.vcf` overlap with a gene? (**Hint:** Use bedtools intersect command)
2. How many SVs found in `ERR1015069.dels.vcf` do not overlap with a gene? (**Hint:** note the -v parameter to bedtools intersect)
3. How many SVs found in `ERR1015069.dels.vcf` overlap with a more strict definition of 50%?
4. What is the closest gene to the structural variant at IV:384220 in `ERR1015069.dels.vcf`?
5. How many SVs overlap between the two files `ERR1015069.dels.vcf` and `ERR1015121.dels.vcf`?
6. How many SVs have a 50% reciprocal overlap between the two files `ERR1015069.dels.vcf` and `ERR1015121.dels.vcf` (**Hint:** first find the option for reciprocal overlap by typing: bedtools intersect -h)

Congratulations, you have reached the end of the tutorial.