Before we get started…

- Clone/download the files in this github repository onto your HPCC account: https://github.com/eleanore-ritter/plb812-svs

- Run the following wherever you downloaded the github repository:
  - `conda env create –f plb812-svs.yml`
  - `conda env create –f plb812-svs-smoove.yml`

If you need any help at all, please ask me ☺

Computationally Decoding **Structural Variants**

Eleanore Ritter

November 16, 2022

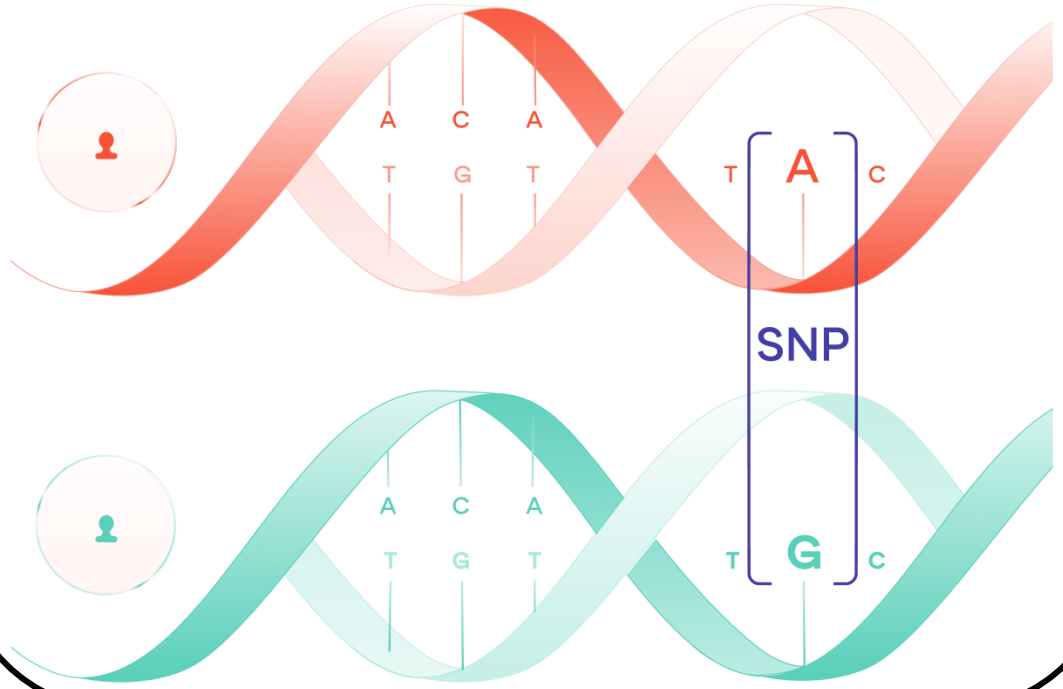
Learning Objectives

- *Know* what structural variants (SVs) are – biologically and computationally!
- *Understand* the strengths and weaknesses of short versus long read sequencing
- *Be able to:*
  - Use short read data to call SVs with DELLY and smoove
  - Pre-process and map long reads
  - Use long read data to call SVs with sniffles and pbsv

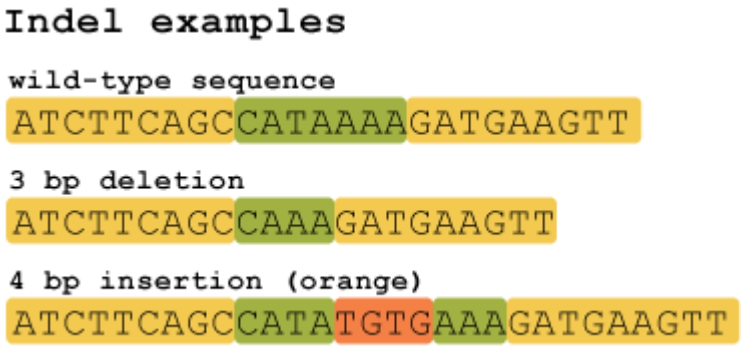
Genomic Variants

Two types of widely studied genomic variants…

Single nucleotide polymorphisms (SNPs)



Insertion or deletion of base pairs (Indels)



Genomic Variants

Improvements in sequencing technologies and cost have enabled the study of larger **structural variants (SVs)**

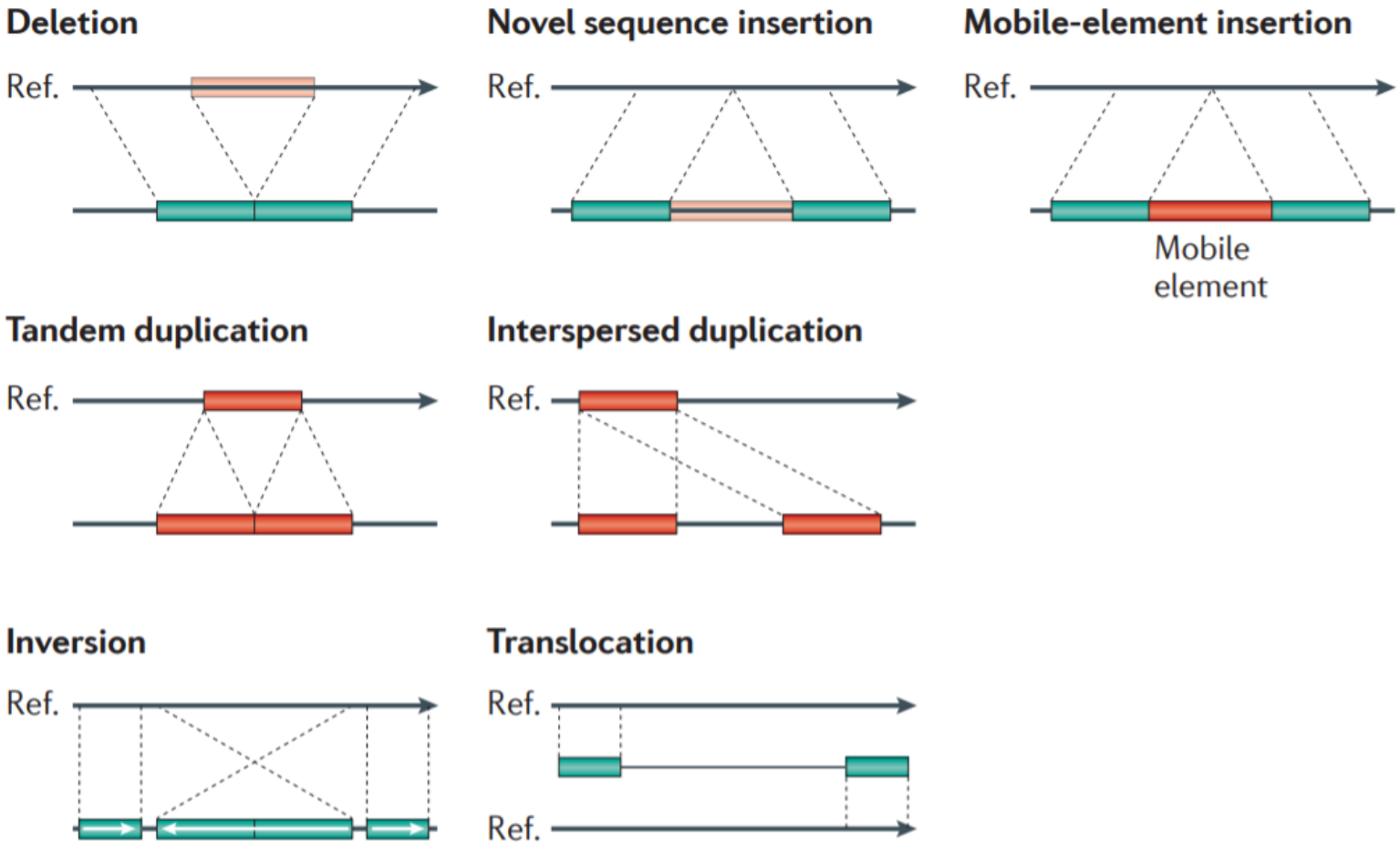


Image from Alkan et al. 2011

Why do we care?

Figure 1 adapted from Vezzulli et al. 2012

A grapevine example from Vezzulli et al. 2012

- The loss of berry color is due to two different SVs in a Myb gene
  - ~4000 kb (chimeric) deletion in Pinot gris
  - ~100 kb deletion in Pinot blanc

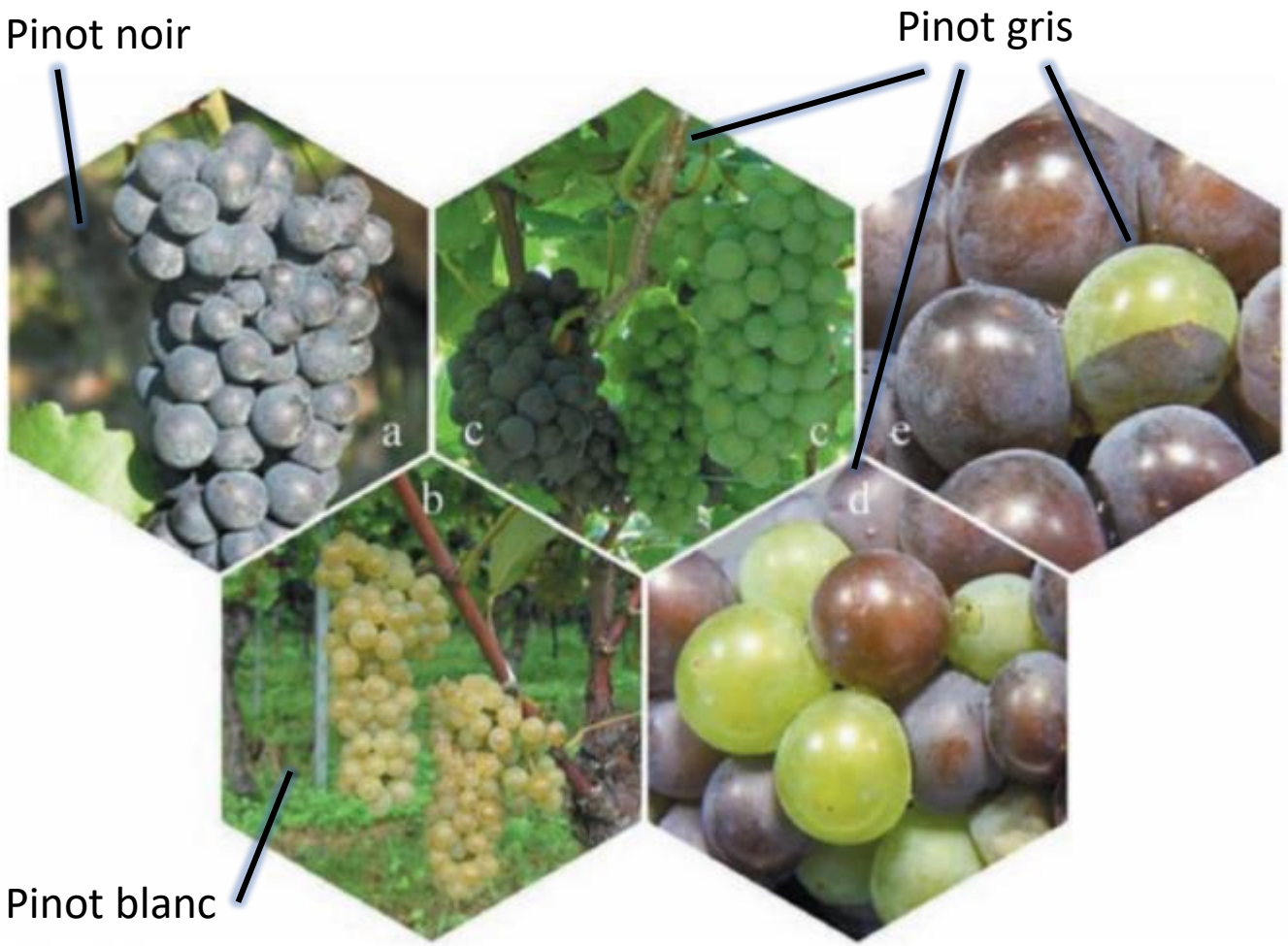


Figure 1 adapted from Vezzulli et al. 2012

The bottom line…

SVs are cool and they impact the plants we study and eat!

USDA ARS

Now, how do we study them?

- Short-read sequencing (ideally to decent coverage, like ≥15X)
- Long-read sequencing (ideally to decent coverage, like ≥15X)
- Comparing genome assemblies

Now, how do we study them?

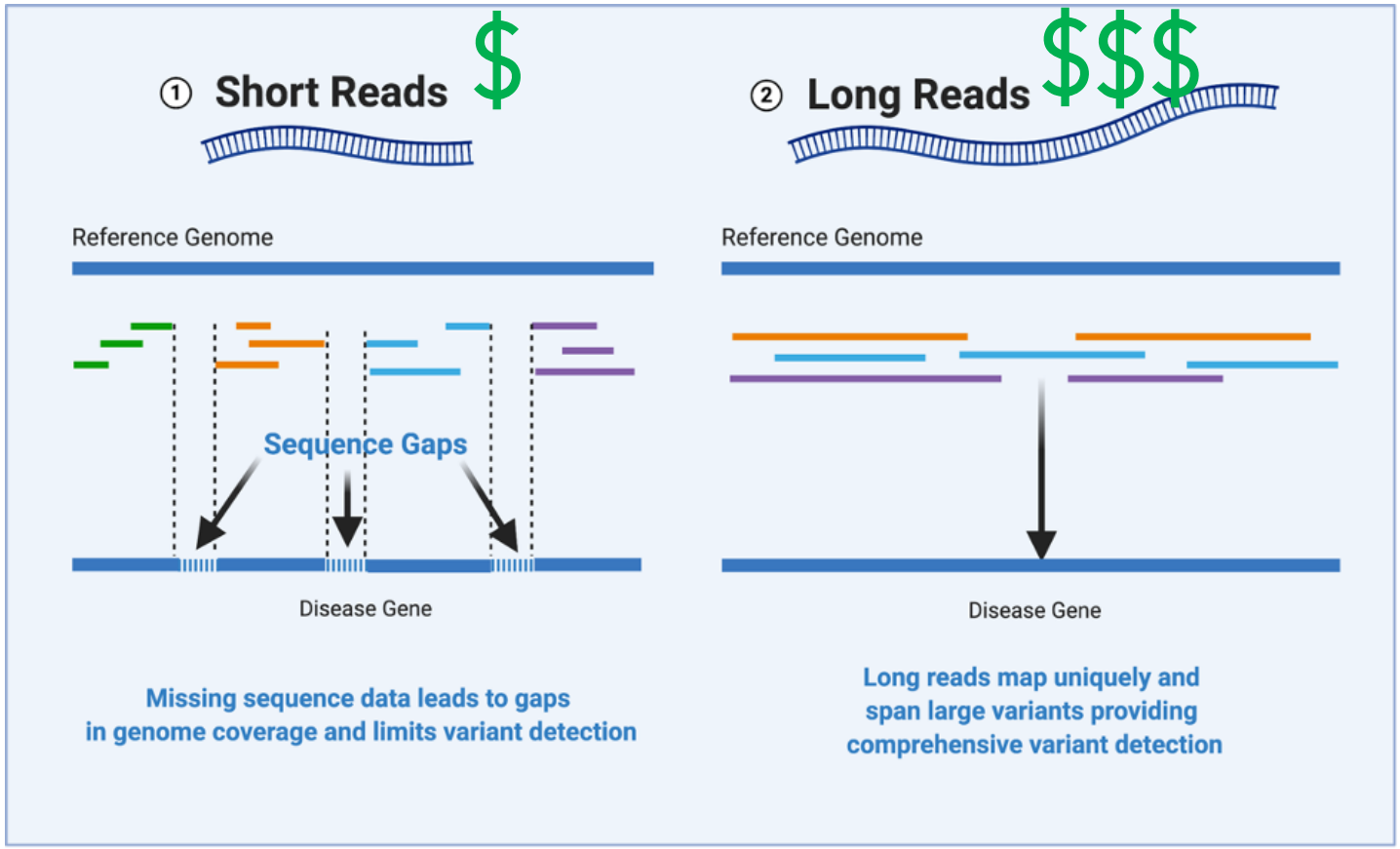
- Short-read sequencing (ideally to decent coverage, ≥15X)
- Long-read sequencing (ideally to decent coverage, ≥ 15X)
- Comparing genome assemblies

Using short- and long-read sequencing data to call SVs is currently more prevalent than comparing genome assemblies due to cost and time, so we will focus on these two.

How do short- and long-reads stack up?

(pun intended)

Short and long reads **both** have their strengths and weaknesses



Large variants are typically easier to accurately call with **long reads**

Illumina NovaSeq 6000 Error Rate: **0.109%** *

Oxford Nanopore Technologies MinION: **6-8%** **

PacBio SMRT cell : **13-15%** (data from older study) ***

*Stoler and Nekruntenko (2021); **Delaheya and Nicolas (2021); ***Ardui et al, (2018)

Image from HudsonAlpha (2021)

Outline for Structural Variant Calling Lesson

Calling SVs with short reads

1. Read processing
2. Mapping
3. Structural variant calling with **DELLY and smoove/lumpy**
4. Filtering
5. Merging
6. Annotating

Calling SVs with long reads

1. Read processing
2. Mapping
3. Structural variant calling with **sniffles and pbsv**
4. Filtering
5. Merging
6. Annotating

Different

Pretty much the same

Walkthrough of studying SVs

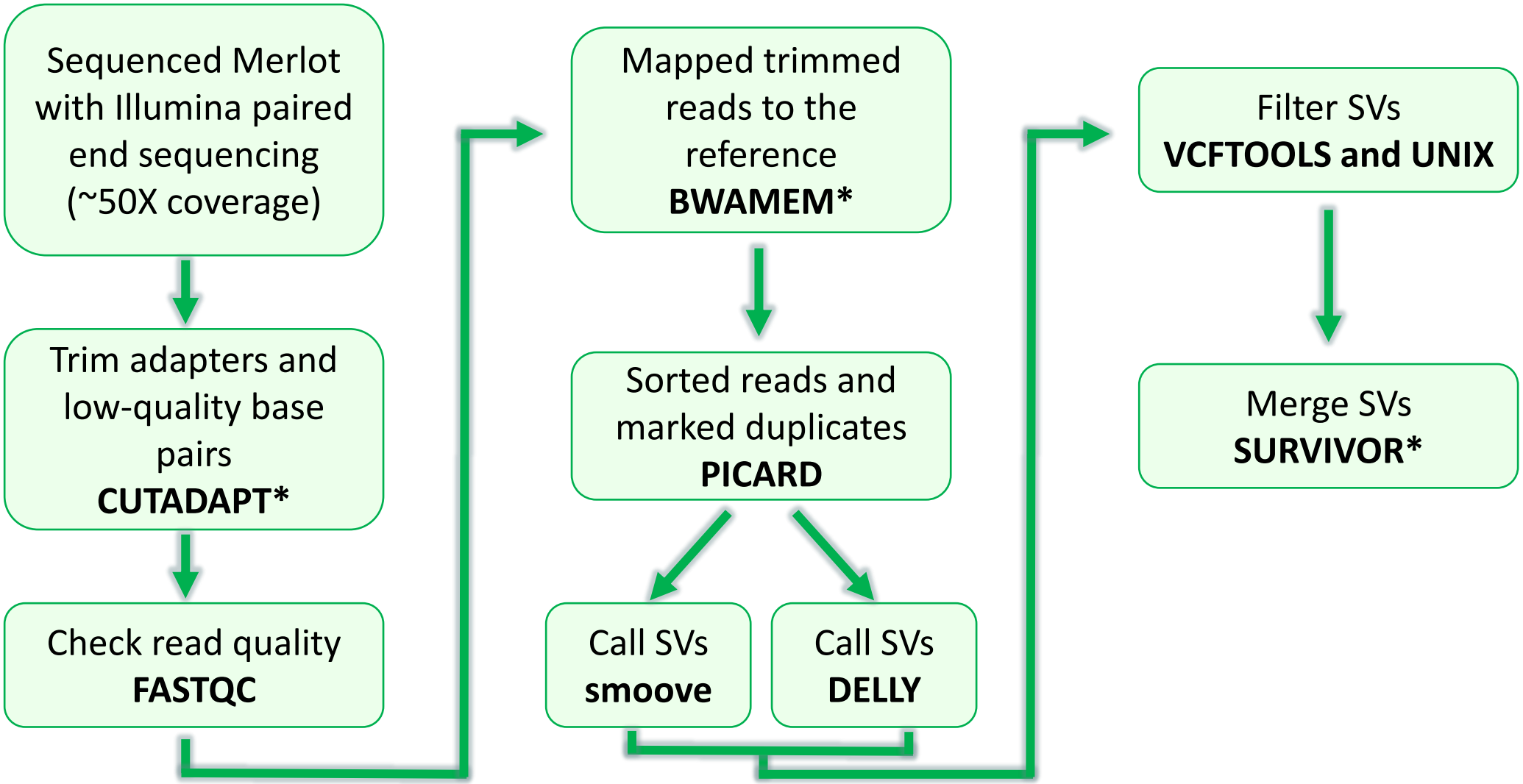
As a case-study, the Merlot variety of grapevine:

Merlot grapes / Getty from winemag.com

Part 1

Calling SVs with short reads

Walkthrough of calling SVs with short reads



* There are alternatives to these programs, and we will talk about them

Actual code in this box

Cutadapt: filters and trims raw reads

Why filter and trim reads?
To remove adapters and low-quality reads that will cause mis-mapping and other issues.

Alternative programs
Trimmomatic – can also be used, more of a personal preference

```
# run cutadapt to clean PE1 read file
# -f file format
# -q trim bases with quality less than 20 from beginning and end of read
# --trim-n trim "N" bases
# -m minimum read length after trimming in 30 bp
# -n remove up to 3 adapters from read ends
# -a Illumina adapter forward read
# -A Illumina adapter reverse read
# -o Output name for read 1
# -p Output name for read 2
cutadapt \
-q 20,20 \
--trim-n \
-m 30 \
-n 3 \
-a AGATCGGAAGAGCACACGTCTGAACTCCAGTCA \
-A AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT \
-o Dakopa_Wild_Type_S3_L002_R1_001.cutadapt.fq.gz Dakopa_Wild_Type_S3_L002_R1_001.fastq.gz \
-p Dakopa_Wild_Type_S3_L002_R2_001.cutadapt.fq.gz Dakopa_Wild_Type_S3_L002_R2_001.fastq.gz
```

Link to manual: https://cutadapt.readthedocs.io/en/stable/

FastQC: check read quality

Why check read quality?
Sometimes issues happen in sequencing, and you need to make sure that your reads are of good quality
(garbage in = garbage out)

```
# run FastQC to check read quality
# -o output directory
# -f file format
# -t threads for running
# input sequences
```

```
fastqc -o /mnt/home/rittere5/ch_rotation/fastqc_cleaned_reads/ \
-f fastq -t 1 Dakopa_Wild_Type_S3_L002_R1_001.cutadapt.fq.gz \
Dakopa_Wild_Type_S3_L002_R2_001.cutadapt.fq.gz
```

Link to manual: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/

BWAMEM: map reads to the reference

We have to map reads to the reference genome to be able to call variants between the reference and the sample.
Mapping programs can impact the ability to call certain variants, and we will talk about that on the next slide. **For that reason, BWAMEM should be used.**

```
# run bwa mem
# -M mark shorter split hits as secondary (for Picard compatability)
# -R complete read group header line
# input reference fasta file
# paired end read files
# > output
bwa mem \
-M \
-R "@RG\tID:$ID\tLB:$LB\tPL:$PL\tSM:$SM\tPU:$PU" \
Vvinifera.fa \
Dakopa_Wild_Type_S3_L002_R1_001.cutadapt.fq.gz Dakopa_Wild_Type_S3_L002_R2_001.cutadapt.fq.gz > Dakopa_WT_bwamem_1.sam
```

Link to manual: http://bio-bwa.sourceforge.net/

Why BWAMEM?

- Most SV callers were designed to work with BWAMEM outputs
  - smoove explicitly states to align with BWAMEM
- Some aligners, like bowtie, do not do split alignment
  - There outputs should not be used for identifying SVs from short reads as a result
- **It is important to pay attention to how your mapping program aligns reads, because this will have a great impact on variant calling**

The **BAM** file

- Binary version of the SAM file
- Tab-delimited text file with sequence alignment data
- Example of a SAM file:

Headers

Alignments

```
1 @HD VN:1.0  SO:unsorted
2 @SQ SN:gi|110640213|ref|NC_008253.1|    LN:4938920
3 @PG ID:bowtie2  PN:bowtie2  VN:2.1.0
4 gi|110640213|ref|NC_008253.1|_418_952_1:0:0_1:0:0_0/1   0   gi|110640213|ref|NC_
  008253.1|   418 42  70M *   0   0   CCAGGCAGTGGCAGGTGGCCACCGTCCTCTCTGCCCCCGCCAAA
  ATCACCAACCATCTGGTAGCGATGAT  22222222222222222222222222222222222222222222222222
  222222222222222222  AS:i:-3 XN:i:0  XM:i:1  XO:i:0  XG:i:0  NM:i:1  MD:Z:8G61
  YT:Z:UU
5 gi|110640213|ref|NC_008253.1|_31_476_0:0:0_0:0:0_1/1    16  gi|110640213|ref|NC_
  008253.1|   407 42  70M *   0   0   GGAAAGCAATGCCAGGCAGGGGCAGGTGGCCACCGTCCTCTCTG
  CCCCCGCCAAAATCACCAACCATCTG  22222222222222222222222222222222222222222222222222
  222222222222222222  AS:i:0  XN:i:0  XM:i:0  XO:i:0  XG:i:0  NM:i:0  MD:Z:70 YT:Z
  :UU
6 gi|110640213|ref|NC_008253.1|_210_743_2:0:0_1:1:0_2/1   0   gi|110640213|ref|NC_
  008253.1|   210 42  70M *   0   0   CATTACCACCACCATCACCATTACCACAGGAAACGGTGCGGGCT
  GACGCGTACAGGAAACACCGAAAAAA  22222222222222222222222222222222222222222222222222
  222222222222222222  AS:i:-6 XN:i:0  XM:i:2  XO:i:0  XG:i:0  NM:i:2  MD:Z:30T31A7
      YT:Z:UU
```

Each row describes a single alignment of a raw read against the reference genome. Each alignment has 11 mandatory fields, followed by any number of optional fields.

Image from
http://biobits.org/samtools_primer.html

Link also contains a helpful description of SAM files

Picard: sort reads and mark duplicates

Why sort reads?
Most downstream programs required sorted reads to be able to more quickly process reads and call variants.

Sort reads:

```
# run picardtools SortSam
# -Xmx19G 19G of memory
# -jar java script to run
# I= input file
# O= output file
# SORT_ORDER= sorting type
java -Xmx19G -jar /mnt/home/rittere5/programs/picard.jar SortSam \
I=Dakopa_WT_bwamem.bam \
O=Dakopa_WT_bwamem_sorted.bam \
SORT_ORDER=coordinate
```

Link to manual: https://broadinstitute.github.io/picard/

Why mark duplicates?
PCR/optical duplicates occur during the sequencing process and can be mistaken as actual reads. For studying SVs, this could cause issues and result in false positives.

Mark duplicates:

```
# run picard markduplicates
# -Xmx19G 19G of memory
# -jar java script to run
# VALIDATION_STRINGENCY= validation stringency for all files
# I= input file
# O= output file
# M= metrics file output
java -Xmx19G -jar /mnt/home/rittere5/programs/picard.jar MarkDuplicates \
VALIDATION_STRINGENCY=LENIENT \
I=Dakopa_WT_bwamem_1_sorted.bam \
O=Dakopa_WT_marked_duplicates.bam \
M=Dakopa_WT_marked_dup_metrics.txt
```

Smoove/LUMPY and DELLY: call SVs

An introduction to the programs:

- Identify novel breakpoints (junctions) between reference and sample
- Analyze alignment signals
  - Increase in read coverage -> potential duplication
  - Decrease in read coverage -> potential deletion
  - Split reads -> potential deletion, translocation, inversion

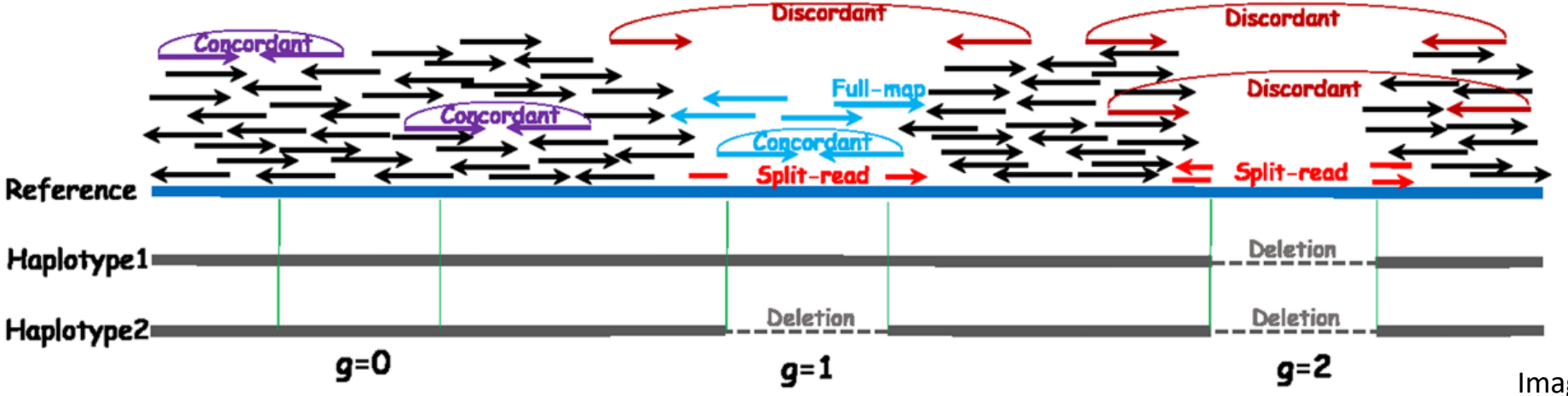


Image from Chu et al. 2014

Smoove/LUMPY and DELLY: call SVs

An introduction to the programs:

- Identify novel breakpoints (junctions) between reference and sample
- Analyze alignment signals
  - Increase in read coverage -> potential duplication
  - Decrease in read coverage -> potential deletion
  - Split reads -> potential deletion, translocation, inversion

smoove integrates multiple signals

DELLY mainly relies on split reads

smoove/LUMPY: call SVs
An introduction

- As mentioned previously, LUMPY analyzes alignment signals:
  - Increase in read coverage -> potential duplication
  - Decrease in read coverage -> potential deletion
  - Split and discordant reads -> potential deletion, translocation, inversion
- The **inputs** for smoove are the mapped reads files and optionally, read files with extracted split reads and discordant reads (all .bam files)
- smoove **outputs** a VCF (variant call format) file

smoove: call and genotype SVs

```
# run smoove call to call SVs
# -x remove PRPOS and PREND tags from INFO
# --name name used in output files
# --fasta reference fasta file
# -p number of processes to parallelize / cpus per task
# --genotype stream output to svtyper for genotyping
# input file(s)
```

```
smoove call -x --name dakapo-pn --fasta $HOME/witchs-broom/refs/new_grape_assembly/genome/Vvinifera.fa \
-p 1 \
--genotype $HOME/witchs-broom/results/2019-01-18-marked-dup-reads/dakapowb/dakapowb_marked_duplicates.bam \
$HOME/witchs-broom/results/2019-01-18-marked-dup-reads/dakapowt/dakapowt_marked_duplicates.bam
```

Link to manual: https://github.com/brentp/smoove

A **VCF** file

- Stores genetic variants
- Tab-delimited text file

A **VCF** file

Header lines

```
##fileformat=VCFv4.3
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=file:///seq/references/1000GenomesPilot-NCBI36.fasta
##contig=<ID=20,length=62435964,assembly=B36,md5=f126cdf8a6e0c7f379d618ff66beb2da,species="Homo sapiens",taxonomy=x>
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, build 129">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM POS      ID         REF   ALT    QUAL  FILTER   INFO                              FORMAT       NA00001         NA00002
20     14370    rs6054257  G     A      29    PASS     NS=3;DP=14;AF=0.5;DB;H2           GT:GQ:DP:HQ  0|0:48:1:51,51  1|0:48:8:51,51
20     17330    .          T     A      3     q10      NS=3;DP=11;AF=0.017               GT:GQ:DP:HQ  0|0:49:3:58,50  0|1:3:5:65,3
20     1110696  rs6040355  A     G,T    67    PASS     NS=2;DP=10;AF=0.333,0.667;AA=T;DB GT:GQ:DP:HQ  1|2:21:6:23,27  2|1:2:0:18,2
20     1230237  .          T     .      47    PASS     NS=3;DP=13;AA=T                   GT:GQ:DP:HQ  0|0:54:7:56,60  0|0:48:4:51,51
20     1234567  microsat1  GTC   G,GTCT 50    PASS     NS=3;DP=9;AA=G                    GT:GQ:DP     0/1:35:4        0/2:17:2
```

Image from https://en.wikipedia.org/wiki/Variant_Call_Format

A **VCF** file

Header lines

```
##fileformat=VCFv4.3
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=file:///seq/references/1000GenomesPilot-NCBI36.fasta
##contig=<ID=20,length=62435964,assembly=B36,md5=f126cdf8a6e0c7f379d618ff66beb2da,species="Homo sapiens",taxonomy=x>
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, build 129">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
                P,Number=1,Type=Integer,Description="Read Depth">
                Q,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM  POS     ID         REF   ALT    QUAL  FILTER   INFO                              FORMAT       NA00001          NA00002
20      14370   rs6054257  G     A      29    PASS     NS=3;DP=14;AF=0.5;DB;H2           GT:GQ:DP:HQ  0|0:48:1:51,51   1|0:48:8:51,51
20      17330   .          T     A      3     q10      NS=3;DP=11;AF=0.017               GT:GQ:DP:HQ  0|0:49:3:58,50   0|1:3:5:65,3
20      1110696 rs6040355  A     G,T    67    PASS     NS=2;DP=10;AF=0.333,0.667;AA=T;DB GT:GQ:DP:HQ  1|2:21:6:23,27   2|1:2:0:18,2
20      1230237 .          T     .      47    PASS     NS=3;DP=13;AA=T                   GT:GQ:DP:HQ  0|0:54:7:56,60   0|0:48:4:51,51
20      1234567 microsat1  GTC   G,GTCT 50    PASS     NS=3;DP=9;AA=G                    GT:GQ:DP     0/1:35:4         0/2:17:2
```

Variants

Image from https://en.wikipedia.org/wiki/Variant_Call_Format

A **VCF** file

```
##fileformat=VCFv4.3
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=file:///seq/references/1000GenomesPilot-NCBI36.fasta
##contig=<ID=20,length=62435964,assembly=B36,md5=f126cdf8a6e0c7f379d618ff66beb2da,species="Homo sapiens",taxonomy=x>
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, build 129">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM  POS      ID         REF   ALT    QUAL  FILTER   INFO                              FORMAT       NA00001          NA00002
20      14370    rs6054257  G     A      29    PASS     NS=3;DP=14;AF=0.5;DB;H2           GT:GQ:DP:HQ  0|0:48:1:51,51   1|0:48:8:51,51
20      17330    .          T     A      3     q10      NS=3;DP=11;AF=0.017               GT:GQ:DP:HQ  0|0:49:3:58,50   0|1:3:5:65,3
20      1110696  rs6040355  A     G,T    67    PASS     NS=2;DP=10;AF=0.333,0.667;AA=T;DB GT:GQ:DP:HQ  1|2:21:6:23,27   2|1:2:0:18,2
20      1230237  .          T     .      47    PASS     NS=3;DP=13;AA=T                   GT:GQ:DP:HQ  0|0:54:7:56,60   0|0:48:4:51,51
20      1234567  microsat1  GTC   G,GTCT 50    PASS     NS=3;DP=9;AA=G                    GT:GQ:DP     0/1:35:4         0/2:17:2
```

Chromosome

Position (bp)

Image from https://en.wikipedia.org/wiki/Variant_Call_Format

A **VCF** file

```
##fileformat=VCFv4.3
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=file:///seq/references/1000GenomesPilot-NCBI36.fasta
##contig=<ID=20,length=62435964,assembly=B36,md5=f126cdf8a6e0c7f379d618ff66beb2da,species="Homo sapiens",taxonomy=x>
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=                ype=Flag,Description="dbSNP membership, build 129">
##INFO=<ID=                ype=Flag,Description="HapMap2 membership">
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM POS      ID        REF   ALT    QUAL  FILTER   INFO                              FORMAT       NA00001         NA00002
20     14370    rs6054257 G     A      29    PASS     NS=3;DP=14;AF=0.5;DB;H2           GT:GQ:DP:HQ  0|0:48:1:51,51  1|0:48:8:51,51
20     17330    .         T     A      3     q10      NS=3;DP=11;AF=0.017               GT:GQ:DP:HQ  0|0:49:3:58,50  0|1:3:5:65,3
20     1110696  rs6040355 A     G,T    67    PASS     NS=2;DP=10;AF=0.333,0.667;AA=T;DB GT:GQ:DP:HQ  1|2:21:6:23,27  2|1:2:0:18,2
20     1230237  .         T     .      47    PASS     NS=3;DP=13;AA=T                   GT:GQ:DP:HQ  0|0:54:7:56,60  0|0:48:4:51,51
20     1234567  microsat1 GTC   G,GTCT 50    PASS     NS=3;DP=9;AA=G                    GT:GQ:DP     0/1:35:4        0/2:17:2
```

ID

Image from https://en.wikipedia.org/wiki/Variant_Call_Format

A **VCF** file

```
##fileformat=VCFv4.3
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=file:///seq/references/1000GenomesPilot-NCBI36.fasta
##contig=<ID=20,length=62435964,assembly=B36,md5=f126cdf8a6e0c7f379d618ff66beb2da,species="Homo sapiens",taxonomy=x>
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Desc
=Flag,Descri
=Flag,Descri
n="Quality b
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM POS      ID         REF  ALT    QUAL  FILTER   INFO                              FORMAT       NA00001         NA00002
20    14370    rs6054257  G    A      29    PASS     NS=3;DP=14;AF=0.5;DB;H2           GT:GQ:DP:HQ  0|0:48:1:51,51  1|0:48:8:51,51
20    17330    .          T    A      3     q10      NS=3;DP=11;AF=0.017               GT:GQ:DP:HQ  0|0:49:3:58,50  0|1:3:5:65,3
20    1110696  rs6040355  A    G,T    67    PASS     NS=2;DP=10;AF=0.333,0.667;AA=T;DB GT:GQ:DP:HQ  1|2:21:6:23,27  2|1:2:0:18,2
20    1230237  .          T    .      47    PASS     NS=3;DP=13;AA=T                   GT:GQ:DP:HQ  0|0:54:7:56,60  0|0:48:4:51,51
20    1234567  microsat1  GTC  G,GTCT 50    PASS     NS=3;DP=9;AA=G                    GT:GQ:DP     0/1:35:4        0/2:17:2
```

Reference allele

Alternate allele
(aka the variant allele)

Image from https://en.wikipedia.org/wiki/Variant_Call_Format

A **VCF** file

```
##fileformat=VCFv4.3
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=file:///seq/references/1000GenomesPilot-NCBI36.fasta
##contig=<ID=20,length=62435964,assembly=B36,md5=f126cdf8a6e0c7f379d618ff66beb2da,species="Homo sapiens",taxonomy=x>
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele
##INFO=<ID=AA,Number=1,Ty  on="Ancest
##INFO=<ID=DB,Number=0,Ty  ="dbSNP me
##INFO=<ID=H2,Number=0,Ty  ="HapMap2
##FILTER=<ID=q10,Descripti  10">
##FILTER=<ID=s50,Description="Less than 50% of samples
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM POS      ID         REF  ALT    QUAL FILTER  INFO                              FORMAT       NA00001         NA00002
20    14370    rs6054257  G    A      29   PASS    NS=3;DP=14;AF=0.5;DB;H2           GT:GQ:DP:HQ  0|0:48:1:51,51  1|0:48:8:51,51
20    17330    .          T    A      3    q10     NS=3;DP=11;AF=0.017               GT:GQ:DP:HQ  0|0:49:3:58,50  0|1:3:5:65,3
20    1110696  rs6040355  A    G,T    67   PASS    NS=2;DP=10;AF=0.333,0.667;AA=T;DB GT:GQ:DP:HQ  1|2:21:6:23,27  2|1:2:0:18,2
20    1230237  .          T    .      47   PASS    NS=3;DP=13;AA=T                   GT:GQ:DP:HQ  0|0:54:7:56,60  0|0:48:4:51,51
20    1234567  microsat1  GTC  G,GTCT 50   PASS    NS=3;DP=9;AA=G                    GT:GQ:DP     0/1:35:4        0/2:17:2
```

Quality of the variant

Filter
(PASS = passed all filters, otherwise failed the filter(s) listed)

Image from https://en.wikipedia.org/wiki/Variant_Call_Format

A **VCF** file

```
##fileformat=VCFv4.3
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=file:///seq/references/1000GenomesPilot-NCBI36.fasta
##contig=<ID=20,length=62435964,assembly=B36,md5=f126cdf8a6e0c7f379d618ff66beb2da,species="Homo sapiens",taxonomy=x>
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, build 129">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 mem
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM POS      ID         REF   ALT    QUAL  FILTER  INFO                              FORMAT       NA00001         NA00002
20    14370    rs6054257  G     A      29    PASS    NS=3;DP=14;AF=0.5;DB;H2           GT:GQ:DP:HQ  0|0:48:1:51,51  1|0:48:8:51,51
20    17330    .          T     A      3     q10     NS=3;DP=11;AF=0.017               GT:GQ:DP:HQ  0|0:49:3:58,50  0|1:3:5:65,3
20    1110696  rs6040355  A     G,T    67    PASS    NS=2;DP=10;AF=0.333,0.667;AA=T;DB GT:GQ:DP:HQ  1|2:21:6:23,27  2|1:2:0:18,2
20    1230237  .          T     .      47    PASS    NS=3;DP=13;AA=T                   GT:GQ:DP:HQ  0|0:54:7:56,60  0|0:48:4:51,51
20    1234567  microsat1  GTC   G,GTCT 50    PASS    NS=3;DP=9;AA=G                    GT:GQ:DP     0/1:35:4        0/2:17:2
```

Info fields (all described in header)

Image from https://en.wikipedia.org/wiki/Variant_Call_Format

A **VCF** file

```
##fileformat=VCFv4.3
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=file:///seq/references/1000GenomesPilot-NCBI36.fasta
##contig=<ID=20,length=62435964,assembly=B36,md5=f126cdf8a6e0c7f379d618ff66beb2da,species="Homo sapiens",taxonomy=x>
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, bu
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM POS      ID         REF   ALT    QUAL  FILTER   INFO                              FORMAT       NA00001         NA00002
20     14370    rs6054257  G     A      29    PASS     NS=3;DP=14;AF=0.5;DB;H2           GT:GQ:DP:HQ  0|0:48:1:51,51  1|0:48:8:51,51
20     17330    .          T     A      3     q10      NS=3;DP=11;AF=0.017               GT:GQ:DP:HQ  0|0:49:3:58,50  0|1:3:5:65,3
20     1110696  rs6040355  A     G,T    67    PASS     NS=2;DP=10;AF=0.333,0.667;AA=T;DB GT:GQ:DP:HQ  1|2:21:6:23,27  2|1:2:0:18,2
20     1230237  .          T     .      47    PASS     NS=3;DP=13;AA=T                   GT:GQ:DP:HQ  0|0:54:7:56,60  0|0:48:4:51,51
20     1234567  microsat1  GTC   G,GTCT 50    PASS     NS=3;DP=9;AA=G                    GT:GQ:DP     0/1:35:4        0/2:17:2
```

Format fields (all described in header) used to genotype and characterize each individual

Image from https://en.wikipedia.org/wiki/Variant_Call_Format

A **VCF** file

```
##fileformat=VCFv4.3
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=file:///seq/references/1000GenomesPilot-NCBI36.fasta
##contig=<ID=20,length=62435964,assembly=B36,md5=f126cdf8a6e0c7f379d618ff66beb2da,species="Homo sapiens",taxonomy=x>
##phasing=partial
##INFO=<ID=NS,Nu
##INFO=<ID=DP,Nu
##INFO=<ID=AF,Nu
##INFO=<ID=AA,Nu
##INFO=<ID=DB,Nu
##INFO=<ID=H2,Nu
##FILTER=<ID=q10
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM POS      ID         REF   ALT    QUAL  FILTER   INFO                              FORMAT      NA00001         NA00002
20    14370    rs6054257  G     A      29    PASS     NS=3;DP=14;AF=0.5;DB;H2           GT:GQ:DP:HQ 0|0:48:1:51,51  1|0:48:8:51,51
20    17330    .          T     A      3     q10      NS=3;DP=11;AF=0.017               GT:GQ:DP:HQ 0|0:49:3:58,50  0|1:3:5:65,3
20    1110696  rs6040355  A     G,T    67    PASS     NS=2;DP=10;AF=0.333,0.667;AA=T;DB GT:GQ:DP:HQ 1|2:21:6:23,27  2|1:2:0:18,2
20    1230237  .          T     .      47    PASS     NS=3;DP=13;AA=T                   GT:GQ:DP:HQ 0|0:54:7:56,60  0|0:48:4:51,51
20    1234567  microsat1  GTC   G,GTCT 50    PASS     NS=3;DP=9;AA=G                    GT:GQ:DP    0/1:35:4        0/2:17:2
```

Some important format fields for us:

GT = Genotype of the individual (0 being reference allele, 1 being alternate allele)

PE = Number of paired end reads supporting the variants

DP = Read depth

Image from https://en.wikipedia.org/wiki/Variant_Call_Format

A **VCF** file

```
##fileformat=VCFv4.3
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=file:///seq/references/1000GenomesPilot-NCBI36.fasta
##contig=<ID=20,length=62435964,assembly=B36,md5=f126cdf8a6e0c7f379d618ff66beb2da,species="Homo sapiens",taxonomy=x>
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, build 129">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM POS      ID         REF   ALT    QUAL  FILTER   INFO                              FORMAT       NA00001         NA00002
20     14370    rs6054257  G     A      29    PASS     NS=3;DP=14;AF=0.5;DB;H2           GT:GQ:DP:HQ  0|0:48:1:51,51  1|0:48:8:51,51
20     17330    .          T     A      3     q10      NS=3;DP=11;AF=0.017               GT:GQ:DP:HQ  0|0:49:3:58,50  0|1:3:5:65,3
20     1110696  rs6040355  A     G,T    67    PASS     NS=2;DP=10;AF=0.333,0.667;AA=T;DB GT:GQ:DP:HQ  1|2:21:6:23,27  2|1:2:0:18,2
20     1230237  .          T     .      47    PASS     NS=3;DP=13;AA=T                   GT:GQ:DP:HQ  0|0:54:7:56,60  0|0:48:4:51,51
20     1234567  microsat1  GTC   G,GTCT 50    PASS     NS=3;DP=9;AA=G                    GT:GQ:DP     0/1:35:4        0/2:17:2
```

Each individual with variants called; column is organized as shown in format column

Image from https://en.wikipedia.org/wiki/Variant_Call_Format

DELLY: call SVs
An introduction

- As previously mentioned, DELLY analyzes **split read** information only
- The **inputs** for DELLY are the mapped reads and the reference genome fasta file

DELLY: call SVs

```
# run delly call
# -o output file
# -g reference genome fasta file
# input files
delly call \
-o delly_dakopawtwb.bcf \
-g $HOME/witchs-broom/refs/new_grape_assembly/genome/Vvinifera.fa \
$HOME/witchs-broom/results/2019-01-18-marked-dup-reads/dakapowt/dakapowt_marked_duplicates.bam \
$HOME/witchs-broom/results/2019-01-18-marked-dup-reads/dakapowb/dakapowb_marked_duplicates.bam
```

Link to manual: https://github.com/dellytools/delly

Activity 1 - Let’s call SVs with short read data

- First, we need to recombine the reference fasta file into one file with the following code:

```
zcat plb812-Vvinifera-half1.fa.gz plb812-Vvinifera-half2.fa.gz > plb812-Vvinifera.fa
```

- Now, modify and run **delly.sh** and **smoove.sh** using the following files I have provided you:
  - **plb812-merlotwt-sr.bam** [our trimmed, mapped reads]
  - **plb812-Vvinifera.fa** [our reference genome]

Take about 10 minutes to do this activity and then we will go over it

Activity 1 Answer for smoove

```
smoove call \
-x \
--name merlotwt \
--fasta plb812-Vvinifera.fa \
-p 1 \
--genotype plb812-merlotwt-sr.bam
```

Activity 1 Answer for DELLY

```
delly call \
-o delly.bcf \
-g plb812-Vvinifera.fa \
plb812-merlotwt-sr.bam
```

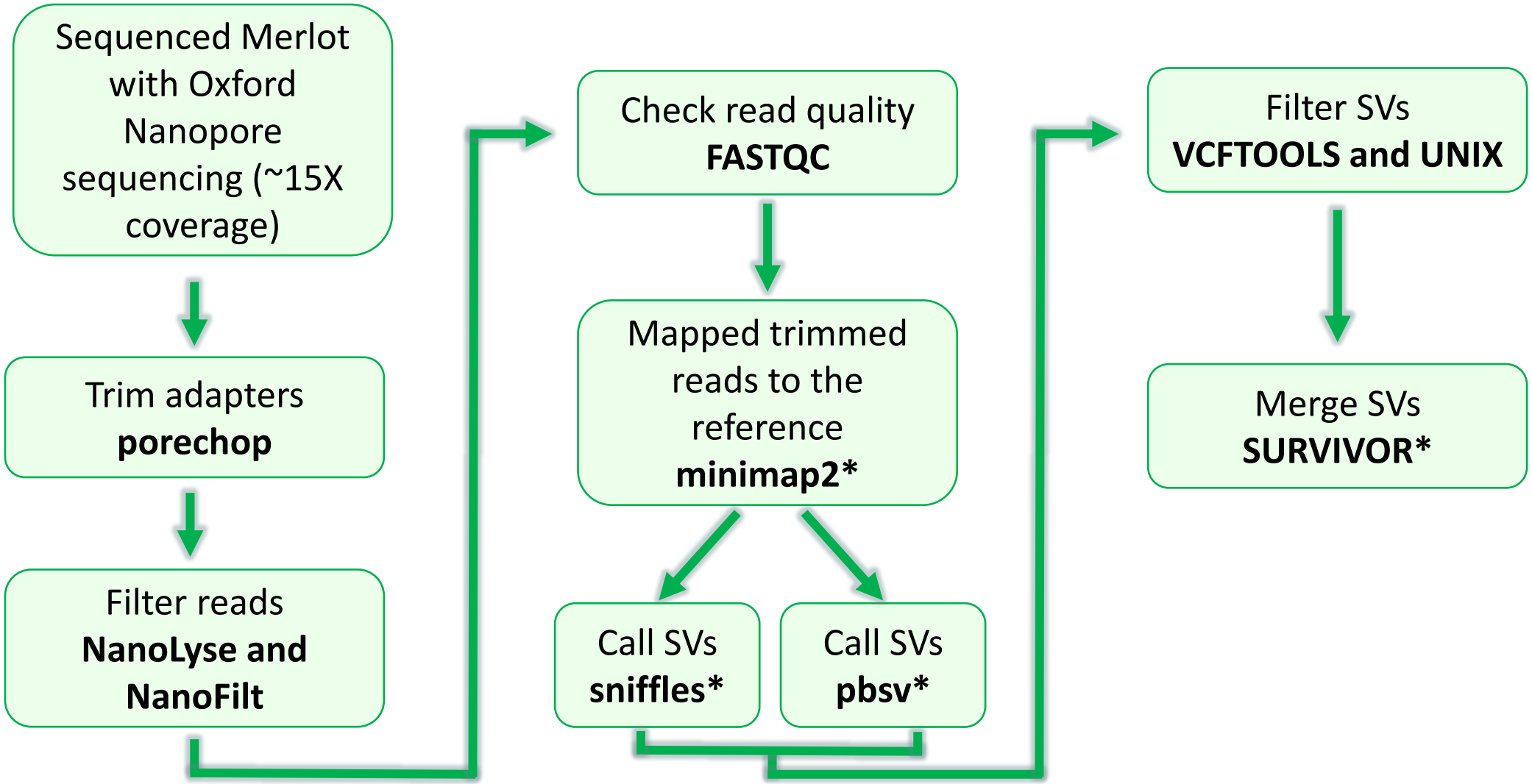
End of Part 1

Any questions?

Part 2

Calling SVs with long reads

Walkthrough of calling SVs with long reads



* There are alternatives to these programs, and we will talk about them

porechop: trim adapters for Nanopore reads

```
# run porechop to trim reads
# -i input fastq.gz file
# -o output trimmed fastq.gz file
# -t threads/ cpus per task to use
# --min_trim_size minimum base pairs needed to algin to adapter
# --extra_end_trim the amount of base pairs to trim past the adapter match to make sure that it is removed
# --end_threshold percent of base pairs required to align
# --middle_threshold percent of base pairs required to align when adapter is found in the middle of a read
# --extra_middle_trim_good_side number of extra bases trimmed after an adapter found in the middle of the read
# --extra_middle_trim_bad_side number of extra bases trimmed before an adapter that is found in the middle of the read
# --min_split_read_size the minimum size to keep of a split read (a read with an adapter in the middle)
```

```
porechop \
-i ${input} \
-o ${output1} \
-t ${threads} \
--min_trim_size 5 \
--extra_end_trim 2 \
--end_threshold 80 \
--middle_threshold 90 \
--extra_middle_trim_good_side 2 \
--extra_middle_trim_bad_side 50 \
--min_split_read_size ${length}
```

Link to manual: https://github.com/rrwick/Porechop#how-it-works

NanoLyse and NanoFilt: filter reads

Why filter reads?
To remove lambda DNA and low-quality reads that will cause mis-mapping and other issues.

```
# Unzip and read input file into NanoLyse
# Run NanoLyse remove reads mapping to the lambda phage genome from a fastq file
# -r lambda reference genome file
# Run NanoFilt to filter reads based on quality and length
# -q filter on a minimum average read quality score
# -l filter on a minimum read length
# zip resulting file and create output file
```

```
zcat ${input} | \
NanoLyse -r ${lambda} | \
NanoFilt -q ${quality} -l ${length} | \
gzip > ${output}
```

Link to manual: https://github.com/wdecoster/nanolyse and https://github.com/wdecoster/nanofilt

FastQC: check read quality

Why check read quality?
Sometimes issues happen in sequencing, and you need to make sure that your reads are of good quality
(garbage in = garbage out)

```
# run FastQC to check read quality
# -o output directory
# -f file format
# -t threads for running
# input sequences
```

```
fastqc -o /mnt/home/rittere5/ch_rotation/fastqc_cleaned_reads/ \
-f fastq -t 1 Dakopa_Wild_Type_S3_L002_R1_001.cutadapt.fq.gz \
Dakopa_Wild_Type_S3_L002_R2_001.cutadapt.fq.gz
```

Link to manual: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/

minimap2: map reads to reference genome

Note
Some SV callers require specific mapping methods or aligners. Be sure to check any manual before mapping with minimap2.

```
# Run minimap2 to generate index
# -x specify datatype (ont, pac, hifi, etc.)
# -d output minimap2 index file
# reference genome fasta file

minimap2 \
-x ${dt} \
-d ${path2}/${ref}.mmi \
${path1}/*.fa

#Run minimap2
echo "Running minimap2"

# Run minimap2 to map reads to reference genome
# -ax sets output as SAM (a) and allows datatype to be preset (x)
# minimap2 index file generated above
# input fastq file - should be trimmed and filtered
# --MD output the MD tag
# -t number of threads to parallelize with
# output sam file

minimap2 \
-ax ${dt} \
${path2}/${ref}.mmi \
${fastq} \
--MD \
-t ${threads} > ${path2}/aln.sam
```

Link to manual: https://github.com/lh3/minimap2

pbsv: call and genotype structural variants

An introduction

pbsv: call and genotype structural variants
An introduction

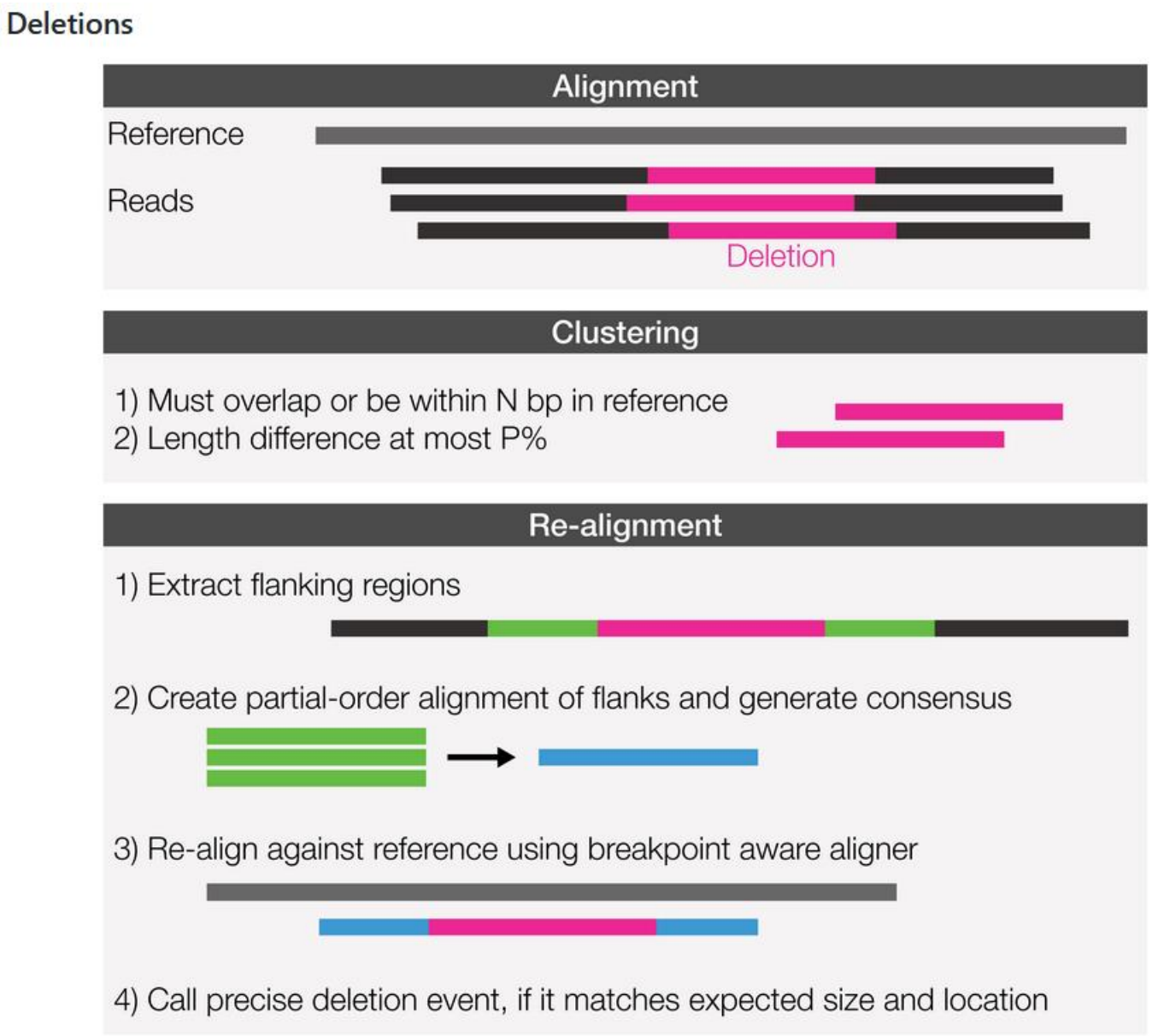


Figure from PacBio https://github.com/PacificBiosciences/pbs

pbsv: call and genotype structural variants
An introduction

Insertions are the same, but with inserted sequence similarity check during clustering

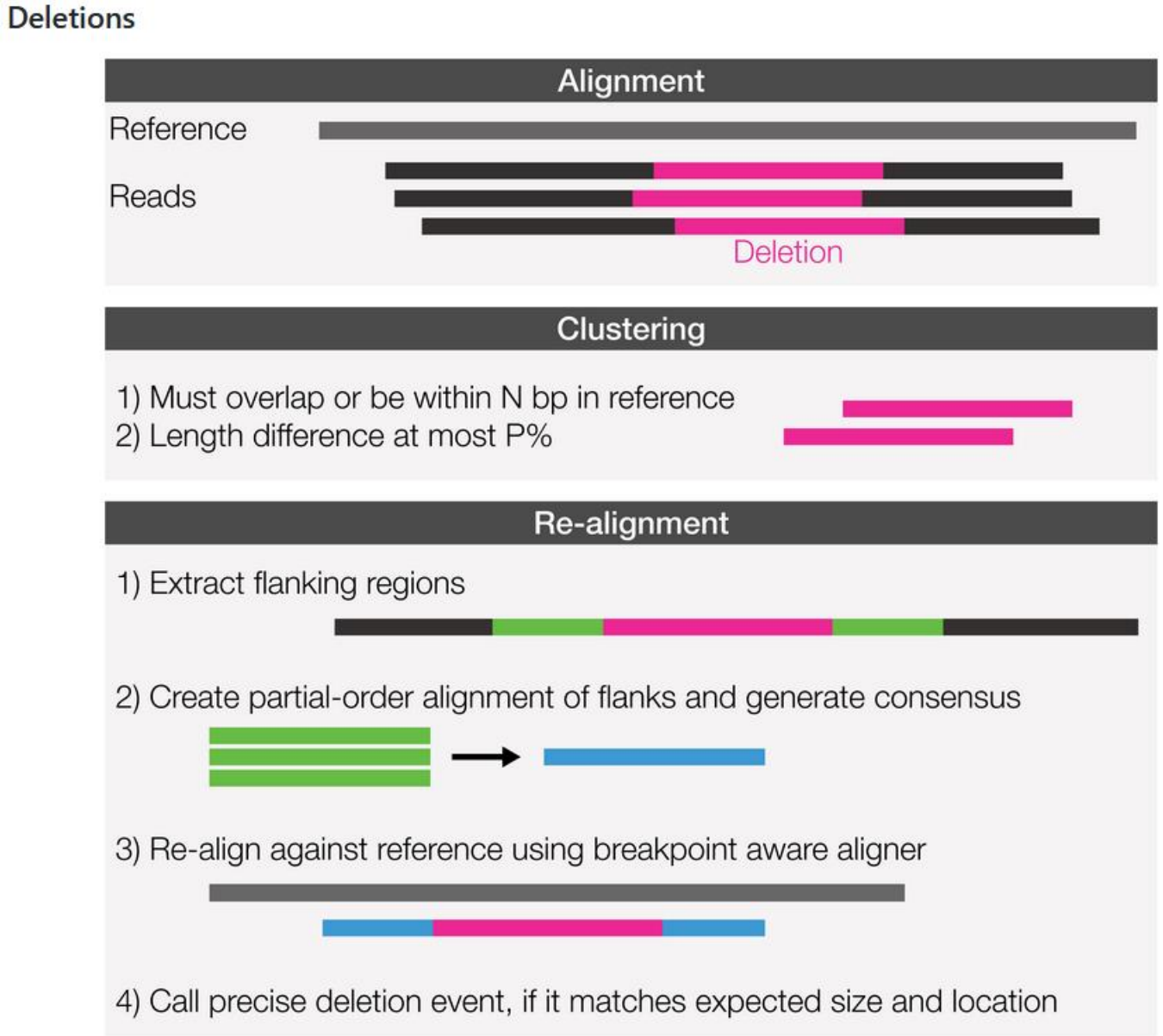


Figure and information from PacBio https://github.com/PacificBiosciences/pbsv

pbsv: call and genotype structural variants
An introduction

Figure and information from PacBio https://github.com/PacificBiosciences/pbsv

pbsv: call and genotype structural variants
An introduction

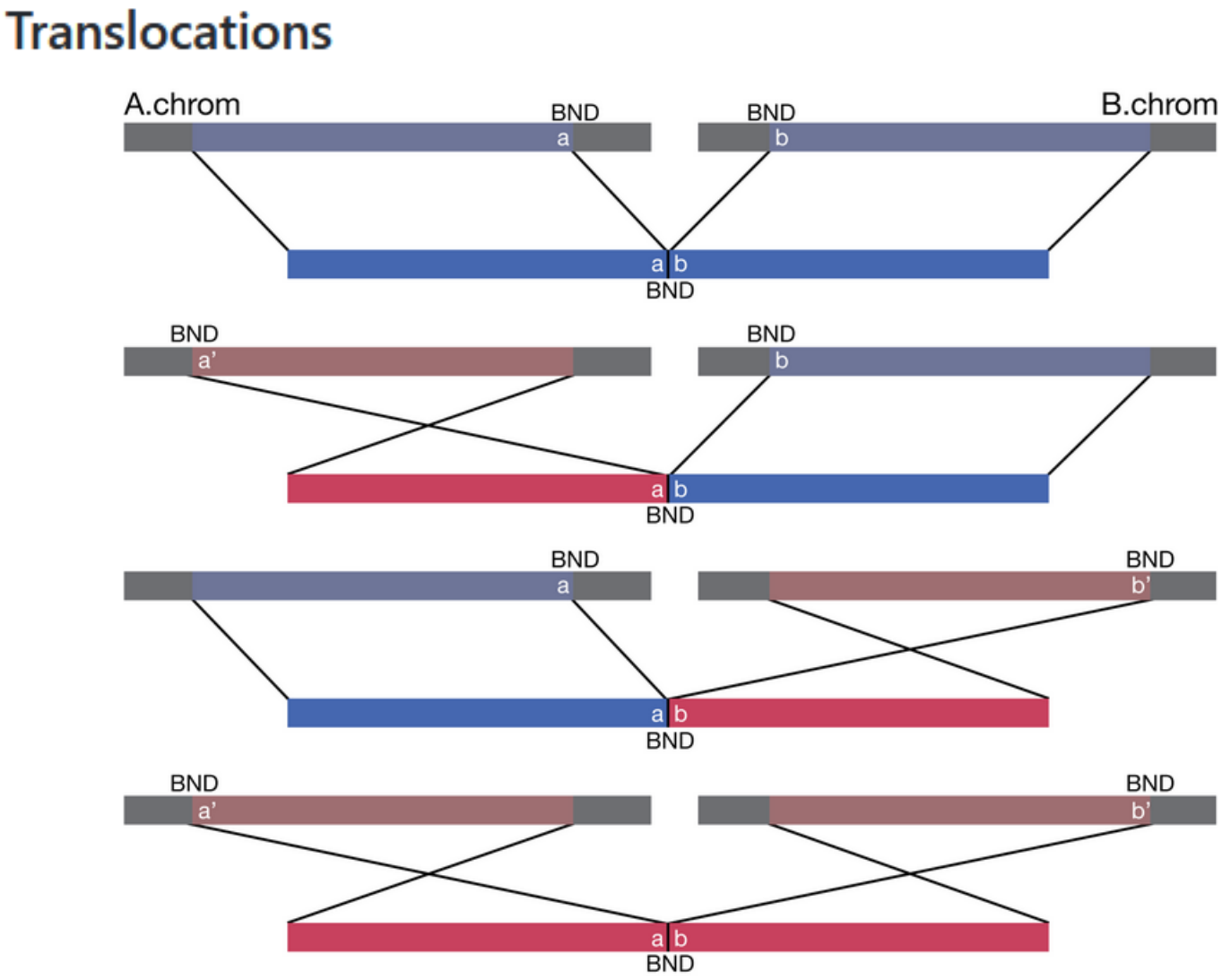


Figure and information from PacBio https://github.com/PacificBiosciences/pbsv

pbsv: call and genotype structural variants
An introduction

Duplications

From split reads

Duplications can be identified from the following split-read signatures:

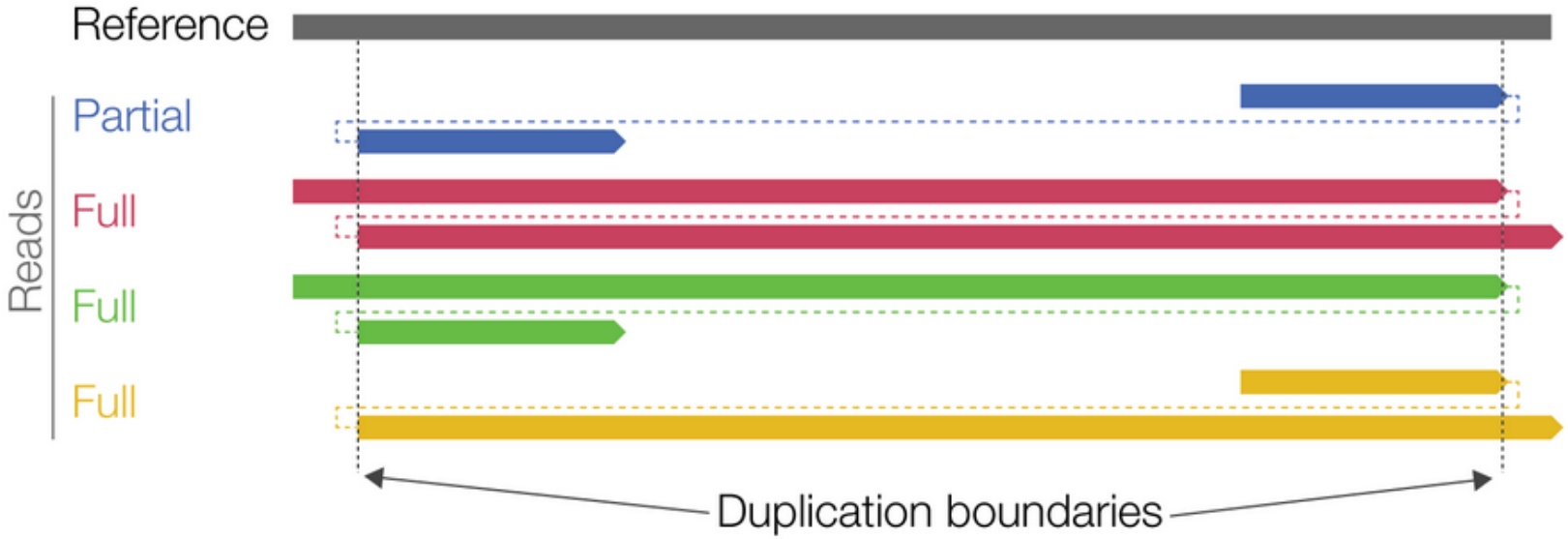


Figure and information from PacBio https://github.com/PacificBiosciences/pbsv

pbsv: call and genotype structural variants

```
#Run pbsv
echo "Running pbsv to discover signatures of structural variation on ${sample1}"

# run pbsv discover to discover signatures of structural variation
# input bam file, sorted and with read groups added
# output svsig.gz file

pbsv discover ../*rg.bam \
${sample1}.svsig.gz

echo "Calling structural variants and assigning genotypes with pbsv"

# run pbsv call to call structural variants and assign genotypes
# reference genome fasta file
# svsig.gz file produced from pbsv discover
# output vcf file

pbsv call \
${path1}/*.fa \
*.svsig.gz \
${sample1}.var.vcf
```

Link to manual: https://github.com/PacificBiosciences/pbsv

sniffles: call and genotype structural variants
An introduction

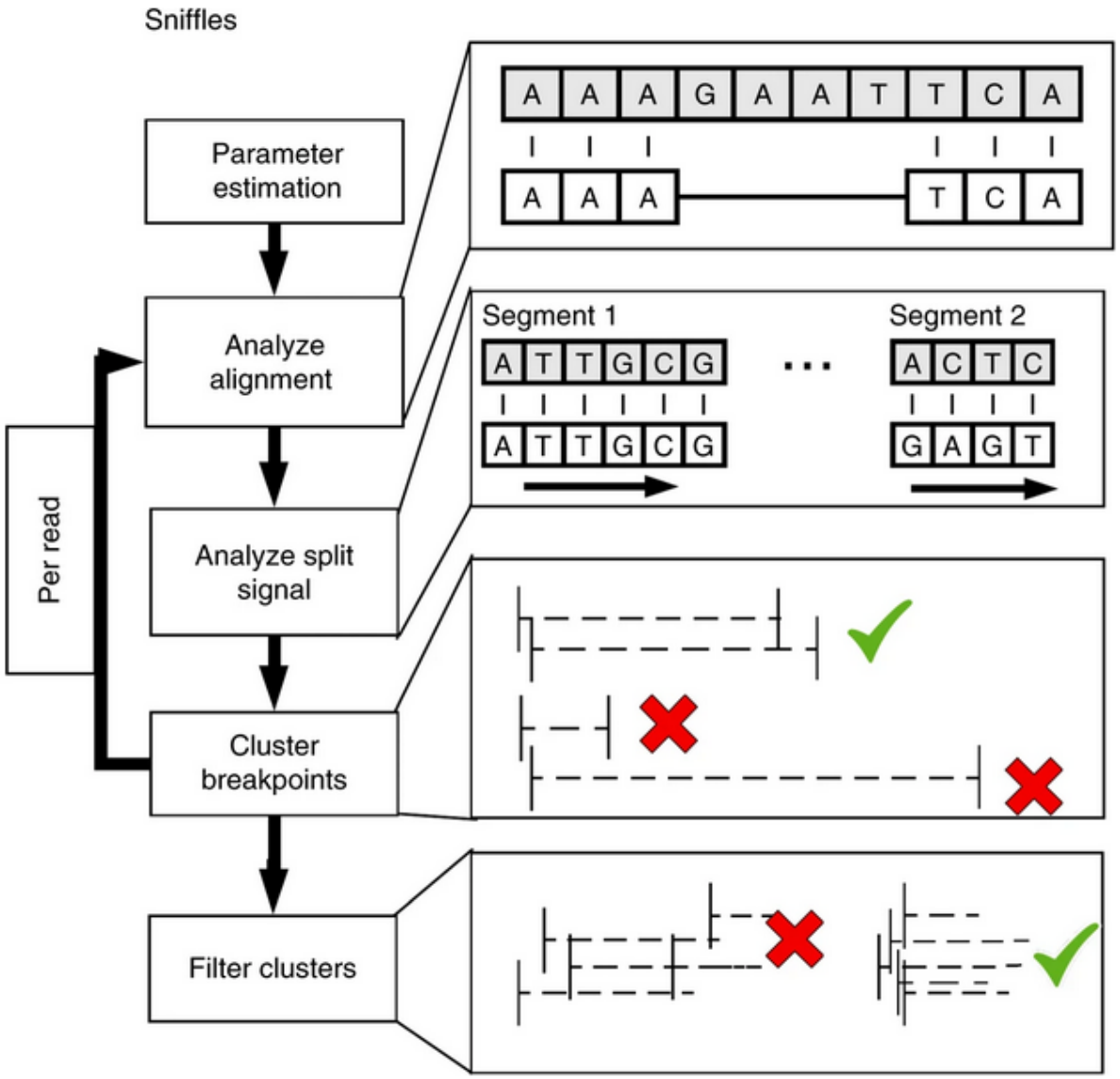


Figure adapted from Sedlazeck et al. 2018

sniffles: call and genotype structural variants

```
# run sniffles to call and genotype SVs
# --input input bam file
# --reference reference genome fasta file
# -v output vcf file
# -t number of threads to use

sniffles --input ../*.bam \
--reference ${path1}/*.fa \
-v ${sample}_sniffles.vcf \
-t ${threads}
```

Link to manual: https://github.com/fritzsedlazeck/Sniffles

Activity 2 - Let’s call SVs with long read data

- First, modify and run **pbsv.sh** using the following files I have provided you:
  - **plb812-merlotwt-ont-pbsv.bam** [our prepped, mapped reads]
  - **plb812-Vvinifera.fa** [our reference genome]

- Next, modify and run sniffles.sh using the following files I have provided you:
  - **plb812-merlotwt-ont-sniffles.bam** [our prepped, mapped reads]
  - **plb812-Vvinifera.fa** [our reference genome]

Take about 10 minutes to do this activity and then we will go over it

Activity 2 Answer for pbsv

```
pbsv discover plb812-merlotwt-ont-pbsv.bam \
merlotwt.svsig.gz

pbsv call \
plb812-Vvinifera.fa \
*.svsig.gz \
merlotwt.var.vcf
```

Activity 2 Answer for sniffles

```
sniffles --input plb812-merlotwt-ont-sniffles.bam \
--reference plb812-Vvinifera.fa \
-v merlotwt_sniffles.vcf \
-t ${threads}
```

End of Part 2

Any questions?

Great, we finished parts 1 and 2 and have the SVs called … Now what?

Part 3

Downstream processing and analysis of VCF files

Filtering variants in VCF files

Filtering is very important!
While many SV callers filter SVs, they do not remove variants that do not pass the filter. We can extract the ones that did pass the filter using UNIX, snpsift, or other programs

- You can use awk in UNIX as a quick and dirty way to filter:

```
awk '$7=="PASS“’ input.vcf # you will need to add the header back
```

- You can also use **snpsift** to filter based on your own prerequisites (read depth, etc.)

Merging variants

- Merging variants identified by **multiple SV callers** can allow us to curate a list of **high-quality SVs**

- Different SV callers have slightly different formats and VCF files with SVs are structured a bit different from SNP VCF files
  - **SURVIVOR** was designed to combine calls from the popular SV callers and can merge all these formats accordingly

- There are many alternatives to SURVIVOR as well

SURVIVOR: merge SVs

```
# run SURVIVOR merge
# file with input file names
# max distance between breakpoints
# min number of supporting caller
# take the type into account (1=yes, 0=no)
# take the strands into account (1=yes, else=no)
# min size of SVs to take into account
# output file name
$HOME/programs/SURVIVOR/Debug/SURVIVOR merge files_dakapo 1000 1 1 0 0 30 dakapowb_merged.vcf
```

Link to manual: https://github.com/fritzsedlazeck/SURVIVOR

Annotating SVs

- Useful if looking for a causal genetic variant
- Can see how many SVs are actually within genes and not intergenic
- Many programs available:
  - **SURVIVOR**
  - **intansv** (in R)
  - and more …
- **Word of caution**: annotations of very large SVs (Mbp long) can be tricky to interpret

Analyzing SVs in R

- **vcfR** is a helpful program to read in VCF files with SVs into R and image variants
  - Capabilities are somewhat limited with SVs, it was built for SNPs and INDELs
- **intansv** is a helpful program to read in SV program outputs and manipulate and image them
  - Similar to SURVIVOR, but includes great plotting functions
  - Version control is extremely important and can cause issues; check https://github.com/venyao/intansv to make sure you are using the right versions of everything

Link to vcfR manual: https://github.com/knausb/vcfR

Link to intansv manual: https://www.bioconductor.org/packages/release/bioc/manuals/intansv/man/intansv.pdf

End of Part 3

Any questions?

Discussion Questions

(or the end of class depending on time)