sideRETRO Documentation

Thiago L. A. Miller

Jan 01, 2020

Contents:

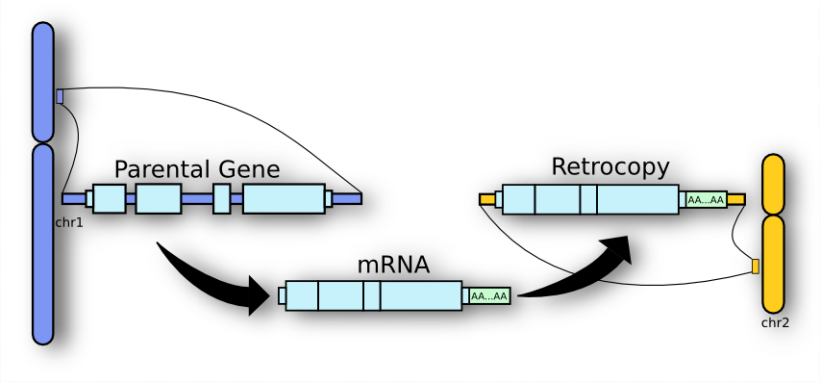
CHAPTER 1

Introduction

sideRETRO is a bioinformatic tool devoted for the detection of somatic (*de novo*) **retrocopy insertion** in whole genome and whole exome sequencing data (WGS, WES). The program has been written from scratch in C, and uses HTSlib and SQLite3 libraries, in order to manage SAM/BAM reading and data analysis. The source code is distributed under the **GNU General Public License**.

1.1 Wait, what is retrocopy?

I can tell you now that retrocopy is a term used for the process resulting from **reverse-transcription** of a mature **mRNA** molecule into **cDNA**, and its insertion into a new position on the genome.



Got interested? For a more detailed explanation about what is a retrocopy at all, please see our section *Retrocopy in a nutshell*.

1.2 Features

When detecting retrocopy mobilization, sideRETRO can annotate several other features related to the event:

Parental gene The **gene** which **underwent retrotransposition** process, giving rise to the retrocopy.

Genomic position The genome **coordinate** where occurred the retrocopy **integration** (chromosome:start-end). It includes the **insertion point**.

Straitens Detects the orientation of the insertion (+/-). It takes into account the orientation of insertion, whether in the **leading** (+) or **lagging** (-) DNA strand.

Genomic context The retrocopy integration site context: If the retrotransposition event occurred at an **intergenic** or **intragenic** region - the latter can be splitted into **exonic** and **intronic** according to the host gene.

Genotype When **multiple** individuals are analysed, annotate the events for each one. That way, it is possible to **distinguish** if an event is **exclusive** or **shared** among the cohort.

Haplotype Our tool provides information about the ploidy of the event, i.e., whether it occurs in one or both **homologous** chromosomes (homozygous or heterozygous).

1.3 How it works

sideRETRO compiles to an executable called `sider`, which has three subcommands: `process-sample`, `merge-call` and `make-vcf`. The `process-sample` subcommand reads a list of SAM/BAM files, and captures **abnormal reads** that must be related to an event of retrocopy. All those data is saved to a **SQLite3 database** and then we come to the second step `merge-call`, which **processes** the database and **annotate** all the retrocopies found. Finally we can run the subcommand `make-vcf` and generate an annotated retrocopy VCF.

```
# List of BAM files
$ cat 'my-bam-list.txt'
/path/to/file1.bam
/path/to/file2.bam
/path/to/file3.bam
...

# Run process-sample step
$ sider process-sample \
  --annotation-file='my-annotation.gtf' \
  --input-file='my-bam-list.txt'

$ ls -1
my-genome.fa
my-annotation.gtf
my-bam-list.txt
out.db

# Run merge-call step
$ sider merge-call --in-place out.db

# Run make-vcf step
$ sider make-vcf \
  --reference-file='my-genome.fa' out.db
```

Take a look at the manual page for *installation* and *usage* information. Also for more details about the algorithm, see our *methodology*.

1.4 Obtaining sideRETRO

The source code for the program can be obtaining in the github page. From the command line you can clone our repository:

```
$ git clone https://github.com/galantelab/sideRETRO.git
```

1.5 No Warranty

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

1.6 Reporting Bugs

If you find a bug, or have any issue, please inform us in the github issues tab. All bug reports should include:

- The version number of sideRETRO
- A description of the bug behavior

1.7 Further Information

If you need additional information, or a closer contact with the authors - *we are always looking for coffee and good company* - contact us by email, see *authors*.

Our bioinformatic group has a site, feel free to make us a visit: https://www.bioinfo.mochsl.org.br/.

CHAPTER 2

Installation

sideRETRO stores its source code on github and uses Meson build system to manage configuration and compilation process.

2.1 Building requirements

- Python 3
- Ninja

The building requirements for **Meson** can be obtained using package manager or from source. For example, using Ubuntu distribution:

```
$ sudo apt-get install python3 \
                       python3-pip \
                       python3-setuptools \
                       python3-wheel \
                       ninja-build
```

2.2 Installing Meson

The recommended way to install the most up-to-date version of **Meson** is through `pip3`:

```
$ pip3 install --user meson
```

For more information about using and installing Meson, see: https://mesonbuild.com/Quick-guide.html

2.3 Project requirements

- zlib

- HTSlib
- SQLite3

If any requirements are not installed, during building, sideRETRO will **download**, **compile** and statically link against the library.

2.4 Compiling and installing

First, you need to **clone** sideRETRO repository:

```
$ git clone https://github.com/galantelab/sideRETRO.git
```

Inside `sideRETRO` folder, **configure** the project with Meson:

```
$ meson build
```

if everything went well, you will see a new **folder** `build`. Now it is time to **compiling** the code:

```
$ ninja -C build
```

It will be created the **executable** `sider` inside the folder `build/src/`, which can already be used. Anyway, if want to **install** sideRETRO to a system folder:

```
$ sudo ninja -C build install
```

By default, sideRETRO will install under `/usr/local/`. The configure script can **customize** the prefix directory. Run:

```
$ meson build configure
```

for instructions and other installation options.

CHAPTER 3

Using sideRETRO

3.1 General Syntax

sideRETRO has a very straightforward syntax. Basically, there are three main commands, each one with a plethora of available options:

- `process-sample`
- `merge-call`
- `make-vcf`

So, in order to test the installation process and run a first example, user can call it without any argument from the command line, like this:

```
$ sider
Usage: sider [-hv]
       sider <command> [options]

A pipeline for detecting
Somatic Insertion of DE novo RETROcopies

Options:
   -h, --help            Show help options
   -v, --version         Show current version

Commands
   ps,  process-sample   Extract alignments related
                         an event of retrocopy
   mc,  merge-call       Discover and annotate
                         retrocopies
   vcf, make-vcf         Generate VCF file with all
                         annotate retrocopies
```

In the above situation, if **sideRETRO** was correctly installed, it will give that default *usage* help.

Another classical example is to print **sideRETRO**'s installed version using the `-v` option:

```
$ sider --version
sideRETRO 0.14.0
```

And, if the user need further help, he can find it both at the **sideRETRO**'s readthedocs page or in the already installed software documentation, from command line:

```
$ sider --help
```

Please, see *A Practical Workflow* and *Running with Docker* sections for more examples and tips for using with **Docker**.

Now, to get more familiar with **sideRETRO** main commands and results, let's try some basic examples for each command.

3.2 Command `process-sample`

The first one is `process-sample` or `ps` for short, and was intended to act as the *"evidence's grounding faith"* for **sideRETRO**. Here, we're saying "first" because of an order in which the user must run the commands. The file resultant from this command will become the input to the next one, `merge-call`.

As explained in the Introduction section, the command `process-sample` creates a database of abnormal reads from a SAM/BAM file set. To do this, there are some mandatory options the user must supply to do a correct search. Calling the command `process-sample` without any argument will give a specific help where user can know all the mandatory options for this command:

```
$ sider process-sample
```

Arguments: One or more alignment file in SAM/BAM format

Mandatory Options:

-a, --annotation-file	Gene annotation on the reference genome in GTF/GFF3 format
-i, --input-file	File containing a newline separated list of alignment files in SAM/BAM format. This option is not mandatory if one or more SAM/BAM files are passed as argument. If 'input-file' and arguments are set concomitantly, then the union of all alignment files is used

Input/Output Options:

-h, --help	Show help options
-q, --quiet	Decrease verbosity to error messages only or suppress terminal outputs at all if 'log-file' is passed
--silent	Same as '–quiet'
-d, --debug	Increase verbosity to debug level
-l, --log-file	Print log messages to a file
-o, --output-dir	Output directory. Create the directory if it does not exist [default:"."]
-p, --prefix	Prefix output files [default:"out"]

SQLite3 Options:

-c, --cache-size	Set SQLite3 cache size in KiB [default:"200000"]

Read Quality Options:

-Q, --phred-quality Minimum mapping quality of the reads required [default:"8"]

-M, --max-base-freq Maximum base frequency fraction allowed [default:"0.75"]

-D, --deduplicate Remove duplicated reads. Reads are considered duplicates when they share the 5 prime positions of both reads and read-pairs

Processing Options:

-s, --sorted Assume all reads are grouped by queryname, even if there is no SAM/BAM header tag 'SO:queryname'

-t, --threads Number of threads [default:"1"]

-m, --max-distance Maximum distance allowed between paired-end reads [default:"10000"]

-f, --exon-frac Minimum overlap required as a fraction of exon [default:"1e-09"; 1 base]

-F, --alignment-frac Minimum overlap required as a fraction of alignment [default:"1e-09"; 1 base]

-e, --either The minimum fraction must be satisfied for at least exon OR alignment. Without '-e', both fractions would have to be satisfied

-r, --reciprocal The fraction overlap must be reciprocal for exon and alignment. If '-f' is 0.5, then '-F' will be set to 0.5 as well

So, supposing that the user has three files: *f1.bam*, *f2.bam*, *f3.sam*, he can type:

```
$ sider process-sample f2.bam f2.bam f3.sam \
    -a annotation_file.gtf
```

Note the mandatory `-a` option specifying the annotation file. And, in this unique exception, we suppressed the `-i` mandatory option cause all the files were explicitly called.

Let's see another example that shows the convenient use of the `-i` option to call a list of input files (e.g. *my_files_list.txt*) instead of them directly:

```
$ sider process-sample \
    -i my_files_list.txt \
    -a annotation_file.gtf
```

Both commands above will produce only one output database file *out.db* containing all relevant reads for non-fixed retrocopies search, whose prefix *out* can be easily changed with the `-p` option. The abnormal reads from all input files will be merged in just one table. To produce one database for each input file separately, user must run one distinct instance of **sideRETRO** per file.

Some options' values can affect drastically the output. Let's play a little bit with some of them while using the short version of the command `ps`:

```
$ sider ps \
    -i my_files_list.txt \
    -a annotation_file.gtf \
    -o output_dir \
    -p my_reads_database \
    -l my_log_file.log \
    -c 2000000 \
    -Q 20 \
    -F 0.9 \
    -t 3
```

Wow! The number of options can be overwhelming.

Here used `-o` option to specify the directory *output_dir* to write our database as *my_reads_database.db* (`-p` option). Also, we chose to save the log messages in *my_log_file.log* file (`-l` option), a cache size of 2Gb (`-c` option), a minimum phred score cutoff of 20 for alignments (`-Q` option), a minimum overlap ratio of 0.9 for read alignments over exonic regions (`-F` option) and 3 threads to process those files in parallel (`-t` option).

To see another example of the `process-sample` command chained in a real workflow, please refer to the *A Practical Workflow* section.

3.3 Command `merge-call`

The second step in the **sideRETRO**'s *"journey for the truth of retrocopies"* is the command `merge-call` or `mc` for short. The aim of this command is to take the database created by `process-sample` step as input and populate more tables in it, with information risen from a clustering process over the abnormal reads regions.

Like `process-sample`, `merge-call` has some mandatory options, which can be known by calling it without any argument:

```
$ sider merge-call
```

Arguments: One or more SQLite3 databases generated in the *process-sample* step

Mandatory Options:

-i, --input-file	File containing a newline separated list of SQLite3 databases to be processed. This option is not mandatory if one or more SQLite3 databases are passed as argument. If 'input-file' and arguments are set concomitantly, then the union of all files is used

Input/Output Options:

-h, --help	Show help options
-q, --quiet	Decrease verbosity to error messages only or suppress terminal outputs at all if 'log-file' is passed
--silent	Same as '–quiet'
-d, --debug	Increase verbosity to debug level
-l, --log-file	Print log messages to a file
-o, --output-dir	Output directory. Create the directory if it does not exist [default:"."]
-p, --prefix	Prefix output files [default:"out"]
-I, --in-place	Merge all databases with the first one of the list, instead of creating a new file

SQLite3 Options:

-c, --cache-size	Set SQLite3 cache size in KiB [default:"200000"]

Clustering Options:

-e, --epsilon	DBSCAN: Maximum distance between two alignments inside a cluster [default:"300"]
-m, --min-pts	DBSCAN: Minimum number of points required to form a dense region [default:"10"]

Filter & Annotation Options:

-b, --blacklist-chr Avoid clustering from and to this chromosome. This option can be passed multiple times [default:"chrM"]

-B, --blacklist-region GTF/GFF3/BED blacklisted regions. If the file is in GTF/GFF3 format, the user may indicate the 'feature' (third column), the 'attribute' (ninth column) and its values

-P, --blacklist-padding Increase the blacklisted regions ranges (left and right) by N bases [default:"0"]

-T, --gff-feature The value of 'feature' (third column) for GTF/GFF3 file [default:"gene"]

-H, --gff-hard-attribute The 'attribute' (ninth column) for GTF/GFF3 file. It may be passed in the format key=value (e.g. gene_type=pseudogene). Each value will match as regex, so 'pseudogene' can capture IG_C_pseudogene, IG_V_pseudogene etc. This option can be passed multiple times and must be true in all of them

-S, --gff-soft-attribute Works as 'gff-hard-attribute'. The difference is if this option is passed multiple times, it needs to be true only once [default:"gene_type=processed_pseudogene tag=retrogene"]

-x, --parental-distance Minimum distance allowed between a cluster and its putative parental gene [default:"1000000"]

-g, --genotype-support Minimum number of reads coming from a given source (BAM) within a cluster [default:"3"]

-n, --near-gene-rank Minimum ranked distance between genes in order to consider them close [default:"3"]

Genotyping Options:

-t, --threads Number of threads [default:"1"]

-Q, --phred-quality Minimum mapping quality used to define reference allele reads [default:"8"]

And likewise, user can call a set of database files directly, or using a list of files:

```
$ sider merge-call database1.db database2.db -I
```

or

```
$ sider merge-call -i my_databases_list.txt -I
```

Note: Again, note the `-I` option that is not mandatory but would lead the creation of duplicated output databases if absent. This option do the clustering "in place" over the input files, overwriting them (so be careful). If user do not use the `-p` or `-I` options, the output files will be named *out.db*.

In a more sophisticated example, we will use the short version of the command `mc`, with many other options:

```
$ sider mc \
    -i my_databases_list.txt \
    -o output_dir \
    -p my_database \
    -l my_log_file.log \
```

(continues on next page)

(continued from previous page)

```
    -I \
    -c 2000000 \
    -B my_black_list.bed \
    -x 1000000 \
    -g 5 \
    -Q 20 \
    -C 15 \
    -t 3
```

Here, options `-i`, `-o`, `-p`, `-l`, `-I`, `-c`, `-Q` and `-t` keeps the same meaning as they have in the `process-sample` command. The others need some explanation. All we've done here was to ask for a minimum number of 5 reads of contribution from each input SAM/BAM file to consider a clustering region as a retrocopy candidate (with `-g` option); a minimum distance of 1000000 bp from the parental gene to resolve some doubtful overlaps (`-x` option), a minimum number of 15 crossing reads over the putative insertion point to consider heterozygosis evidence (`-C`) and, importantly, a BED file with a list of regions to be ignored at the clustering process called *my_black_list.txt* (`-B` option). This last option's file can describe entire chromosomes (like chrM) and many chromosomal regions with poor insertion evidences taken literature, like centromers. All specified regions won't be targets for clustering.

To see another example of the `merge-call` command chained in a real workflow, please refer to the *A Practical Workflow* section.

3.4 Command `make-vcf`

The third and last step to the **sideRETRO**'s *"crusade to retrocopies"* is the `make-vcf` command or `vcf` for short. This command takes the already clustered tables in the database files populated at the `merge-call` step and creates one VCF file with all statistically significant retrocopy insertions annotated in a convenient format.

This command has no mandatory options, but it is worth try to discover the others:

```
$ sider make-vcf
```

Arguments: SQLite3 database generated at *process-sample* and *merge-call* steps

Input/Output Options:

-h, --help	Show help options
-q, --quiet	Decrease verbosity to error messages only or suppress terminal outputs at all if 'log-file' is passed
--silent	Same as '–quiet'
-d, --debug	Increase verbosity to debug level
-l, --log-file	Print log messages to a file
-o, --output-dir	Output directory. Create the directory if it does not exist [default:"."]
-p, --prefix	Prefix output files [default:"out"]

Filter & Annotation Options:

-n, --near-gene-dist	Minimum distance between genes in order to consider them close [default:"10000"]
-e, --orientation-error	Maximum error allowed for orientation rho [default:"0.05"]
-r, --reference-file	FASTA file for the reference genome

So, in order to produce a VCF file from a database input file like *my_database.db*, just type:

```
$ sider make-vcf my_database.db
```

This will produce a *out.vcf* output file.

Let's add more options to customize it to our needs (with the short version of the command only for symmetry):

```
$ sider vcf my_database.db \
    -o output_dir \
    -p my_retrocopies \
    -l my_log_file.log \
    -r my_reference_genome.fa \
    -n 50000
```

Command `make-vcf` is very simple and don't allow the user to use threads. The only new options are `-r`, which must specify the reference genome in FASTA format (like **gencode**'s *Hg38.fa*) and `-n`, where user can establish a distance threshold for genes surrounding insertion points for additional information in the output VCF file.

3.5 A Practical Workflow

Now, let's do an interesting exercise, with real experimental data from the 1000 Genomes Project.

In order to run **siderRETRO** searching for retrocopies, we will download 2 whole-genome sequenced CRAM files, both aligned on the **gencode**'s hg38 genome: NA12878 and NA12778.

While **sideRETRO** can't deal with CRAN files, we'll need to convert them using samtools. And, some steps will require additional files, so at the beginning of a run, the files listed bellow must be at the same directory where the user is running **sideRETRO** or their correct paths must be supplied at the correspondent option. Files are:

1. A GTF gene annotation file from gencode project (here `gencode.v32.annotation.gtf`).
2. A custom BED file to serve as black list – genomic regions to be ignored (here `black_list.bed`).
3. A FASTA file with the gencode's Human reference genome, version 38 (here `Hg38.fa`).
4. A custom perl script, `analyser.pl`, to do the final analysis over the VCF file and produce the TSV file in a tabular format. The `analyser.pl` script can be downloaded `here`.

See the complete command sequence bellow for the whole analysis:

```
# Do things inside a clean directory.
# Average time: irrelevant
$ mkdir -p sider_test
$ cd sider_test

# Create a download list (WGS.list) containing all files of interest.
# Average time: irrelevant
$ echo "ftp://ftp.sra.ebi.ac.uk/vol1/run/ERR323/ERR3239334/NA12878.final.cram" > WGS_
↪download.list
$ echo "ftp://ftp.sra.ebi.ac.uk/vol1/run/ERR323/ERR3239484/NA12778.final.cram" >> WGS_
↪download.list

# Download all files: NA12878 and NA12778.
# Average time: network dependent
$ wget -c -i WGS_download.list

# Convert CRAM file format to BAM using samtools view.
```

(continues on next page)

(continued from previous page)

```
# Average time: 62m34.541
$ samtools view -b -@ 8 -o NA12878.final.bam NA12878.final.cram
$ samtools view -b -@ 8 -o NA12778.final.bam NA12778.final.cram

# Create the list of BAM files.
# Average time: irrelevant
$ echo "*.bam" > WGS_genomes.list

# First sideRETRO step: process-sample
# Input file: WGS_genomes.list
# Output file: 1000_genomes.db
# Average time: 62m34.541
$ sider process-sample \
    -i WGS_genomes.list \
    -a gencode.v32.annotation.gtf \
    -p 1000_genomes \
    -c 2000000 \
    -Q 20 \
    -F 0.9 \
    -t 2

# Second sideRETRO step: merge-call
# Input file: 1000_genomes.db
# Output file: 1000_genomes.db (same file)
# Average time: 62m34.541
$ sider merge-call 1000_genomes.db \
    -c 2000000 \
    -x 1000000 \
    -g 5 \
    -B black_list.bed \
    -I \
    -t 2

# Second sideRETRO step: merge-call
# Input file: 1000_genomes.db
# Output file: 1000_genomes.vcf
# Average time: 62m34.541
$ sider make-vcf 1000_genomes.db \
    -p 1000_genomes \
    -r Hg38.fa

# Some analysis over the final VCF file.
# Input file: 1000_genomes.vcf
# Output file: 1000_genomes.tsv
# Average time: 62m34.541
$ perl analyser.pl 1000_genomes.vcf > 1000_genomes.tsv
```

This was a simple but complete pipeline to obtain a final TSV file with all the relevant results in a tabular format ready to import in a R or Python script and plot some graphics.

In order to compare, the resultant VCF file shown these general statistics:

- 207 lines without headers.
- 26 non-fixed and distinct insertions of retrocopies.
- 14 of them shared among the two genomes.
- 0 of them in homozygosis.

3.6 Running with Docker

Notwithstanding **sideRETRO**'s native run, user can happily run it from a **Docker** image just prepending **Docker**'s directives to any example shown. That is, supposing the user has *Docker* installed and has pulled the image *galantelab/sider:latest* from DockerHub, he can just prepend `docker --rm -ti -v $(pwd):/home/sider -w /home/sider galantelab/sider` to the ordinary `sider` command, like:

```
$ docker --rm -ti -v $(pwd):/home/sider -w /home/sider galantelab/sider \
  sider ps \
      -i my_files_list.txt \
      -a annotation_file.gtf \
      -o output_dir \
      -p my_reads_database \
      -l my_log_file.log \
      -c 2000000 \
      -Q 20 \
      -F 0.9 \
      -t 3
```

CHAPTER 4

Retrocopy in a nutshell

In the late 1940s, Barbara McClintock discovered the controlling elements, later known as transposons[1].

Fig. 1: Image of Barbara McClintock. Cold Spring Harbor Laboratory Archives. Copyright © 2016 by the Genetics Society of America

These elements, also called transposable elements (TEs), collectively comprise more than half of mammals' genome[2] and for humans, approximately two-thirds of the 3 billion base pair genome are the outcome of TEs activity[3]. TEs are subdivided in DNA-transposons and retrotransposons, and the latter being the result of retrotransposition process[4][5]. Those classes of TEs can be autonomous or non-autonomous according to the presence or absence of their own enzymatic machinery of (retro)transposition, respectively. In retrotransposons, the most prominent autonomous elements are LINEs (Long Interspersed Nuclear Elements), and from the non-autonomous class, they are SINEs (Short Interspersed Nuclear Elements) together with processed pseudogenes or retrocopies of mRNAs (retrotransposed protein-coding genes).

[1] MCCLINTOCK, B. (1950). The origin and behavior of mutable loci in maize. Proceedings of the National Academy of Sciences of the United States of America, 36(6), 344–355.

[2] BURNS, K. H. (2017). Transposable elements in cancer. Nature reviews. Cancer, 17(7), 415–424.

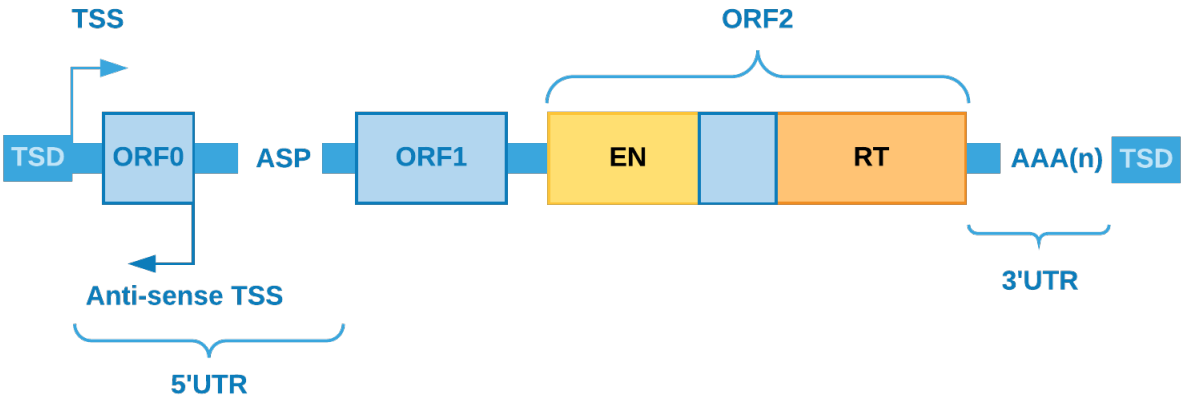
[3] DE KONING, A. P. J. et al. (2011). Repetitive Elements May Comprise Over Two-Thirds of the Human Genome. PLoS genetics, 7(12), e1002384.

[4] KAESSMANN, H. (2010). Origins, evolution, and phenotypic impact of new genes. Genome research, 20(10), 1313–1326.

[5] HELMAN, E. et al. (2014). Somatic retrotransposition in human cancer revealed by whole-genome and exome sequencing. Genome research, 24(7), 1053–1063.

4.1 LINE

LINEs became the most frequent transposable element, in number of nucleotides, corresponding to approximately 17% of the human genome[6]. In our genome, the most numerous family of LINEs is LINE-1 (L1) and when its sequence is full-length (about 6 kb), this element has: i) one promoter region; ii) a 5'UTR region; iii) two coding regions (ORF1p and ORF2p); iv) a 3'UTR region; v) a poly-A tail inside its transcript; vi) and recently a distinct ORF (ORF0, which is 70 amino acids in length, but still with unknown function) was found in primates[7][8].



ORF1p encodes a RNA-binding protein, responsible for the mRNA binding specificity, and ORF2p encodes a dual function protein working as reverse transcriptase and endonuclease. Together, the coding regions of L1s are accountable for shaping the retrotransposase and this machinery can operate in cis making retrocopies of the element itself, or in trans retrocopying non-autonomous repetitive elements, like SINEs and mRNAs transcripts[9][10]. In this process, from mRNA, a cDNA is generated (by retrotranscription) and then randomly inserted back to the nuclear genome, giving birth to a (retro)copy from the original/parental element.

4.2 SINE

SINEs, one of the elements retrotransposed by L1 retrotransposase, account for approximately 11% of the human genome and its most frequent family is Alu with average length of 300bp[11]. Alu is a primate-specific element and has (when in full-length mode) 5' end with internal hallmarks of RNA polymerase III linked by an A-rich region to a 3' end with an oligo-dA-rich sequence that acts as target to the reverse transcription[12]. As well as SINEs, retrocopies of coding genes depend on L1 machinery and they are one of the major sources of de novo genetic variations[13], potentially contributing also to genetic diseases[14]. Nowadays, we know that retrotransposition events are very frequent in many organisms, with more than 1 million copies of Alu[9] and more than 7,800 retroduplication events of coding genes in our genome[15][16].

[6] LANDER, E. S. et al. (2001). Initial sequencing and analysis of the human genome. Nature, 409(6822), 860–921.
[7] HANCKS, D. C. and KAZAZIAN, H. H. (2016). Roles for retrotransposon insertions in human disease. Mobile DNA, 7(9).
[8] DENLI, A. M. et al. (2015). Primate-specific ORF0 contributes to retrotransposon-mediated diversity. Cell, 163(3), 583–593.
[9] BATZER and DEININGE. (2002). Alu repeats and human genomic diversity. Nature reviews. Genetics, 3(5), 370–379.
[10] KAESSMANN, H. et al. (2009). RNA-based gene duplication: mechanistic and evolutionary insights. Nature reviews. Genetics, 10(1), 19–31.
[11] DEININGER, P. (2011). Alu elements: know the SINEs. Genome biology, 12(12), 236.
[12] BAKSHI et al. (2016). DNA methylation variation of human-specific Alu repeats. Epigenetics: official journal of the DNA Methylation Society, 11(2), 163–173.
[13] BECK et al. (2010). LINE-1 retrotransposition activity in human genomes. Cell, 141(7), 1159–1170.
[14] LEE, E. et al. (2012). Landscape of somatic retrotransposition in human cancers. Science, 337(6097), 967–971.
[15] NAVARRO, F. C. P. and GALANTE, P. A. F. (2013). RCPedia: a database of retrocopied genes. Bioinformatics, 29(9), 1235–1237.
[16] NAVARRO, F. C. P. and GALANTE, P. A. F. (2015). A Genome-Wide Landscape of Retrocopies in Primate Genomes. Genome biology and evolution, 7(8), 2265–2275.

4.3 Retrocopy and diseases

In somatic cells, retrotransposition events are repressed by post-transcriptional and epigenetics modifications, but the temporary loss of these controls can lead to new insertions resulting in structural modifications accountable for diseases, as colorectal and lung cancers[17][18][19]. Recently, some authors showed that, in tumorigenic process, there is a strong correlation between colorectal cancer (CRC) progression and the loss of methylation in regions containing LINEs, from the most methylated (normal mucosa) to the least methylated (CRC metastasis), suggesting that LINEs could act as an important marker for CRC progression[20][21]. Alu elements are also rich in CpG residues and, as in LINEs, the methylation of these elements appears to decrease in many tumors contributing to the development of diseases by either altering the expression of some genes in several ways, disrupting a coding region or splice signal[11]. In 2016, Clayton et al.[22] showed a potentially tumorigenic Alu insertion in the enhancer region of the tumor suppressor gene CBL in a breast cancer sample[22]. However, although many studies have highlighted Alu elements as sources of genetic instability and their contribution to carcinogenesis[23][24], other high throughput studies have hidden Alu elements due to the difficulties in developing efficient methods to identify these elements in a tumorigenic context[11]. Retrocopies were also described in tumorigenic context, as the classical case of PTEN and its retrocopy PTEN1[25]. In this paper, Poliseno and others show the critical consequences of the interaction between PTEN and PTENP1, where the retrocopy (pseudogene) is active, regulates coding gene expression by regulating cellular levels of PTEN and is also selectively deleted in cancer. Therefore, finding these retrotranscribed elements became very important in understanding their potential functions in tumorigenesis and tumor heterogeneity.

4.4 References and Further Reading

[17] MIKI, Y. et al. (1992). Disruption of the APC gene by a retrotransposal insertion of L1 sequence in a colon cancer. Cancer research, 52(3), 643–645.

[18] SOLYOM, S. et al. (2012). Extensive somatic L1 retrotransposition in colorectal tumors. Genome research, 22(12), 2328–2338.

[19] COOKE, S. L. et al. (2014). Processed pseudogenes acquired somatically during cancer development. Nature communications, 5, 3644.

[20] SUNAMI, E. et al. (2011). LINE-1 hypomethylation during primary colon cancer progression. PloS one, 6(4), e18884.

[21] HUR, K. et al. (2014). Hypomethylation of long interspersed nuclear element-1 (LINE-1) leads to activation of proto-oncogenes in human colorectal cancer metastasis. Gut, 63(4), 635–646.

[22] CLAYTON, E. A. et al. (2016). Patterns of Transposable Element Expression and Insertion in Cancer. Frontiers in molecular biosciences, 3, 76.

[23] DEININGER, P. L. and BATZER, M. A. (1999). Alu repeats and human disease. Molecular genetics and metabolism, 67(3), 183–193.

[24] BELANCIO et al. (2010). All y'all need to know 'bout retroelements in cancer. Seminars in cancer biology, 20(4), 200–210.

[25] POLISENO, L. et al, (2010). A coding-independent function of gene and pseudogene mRNAs regulates tumour biology. Nature, 465(7301), 1033–1038.

CHAPTER 5

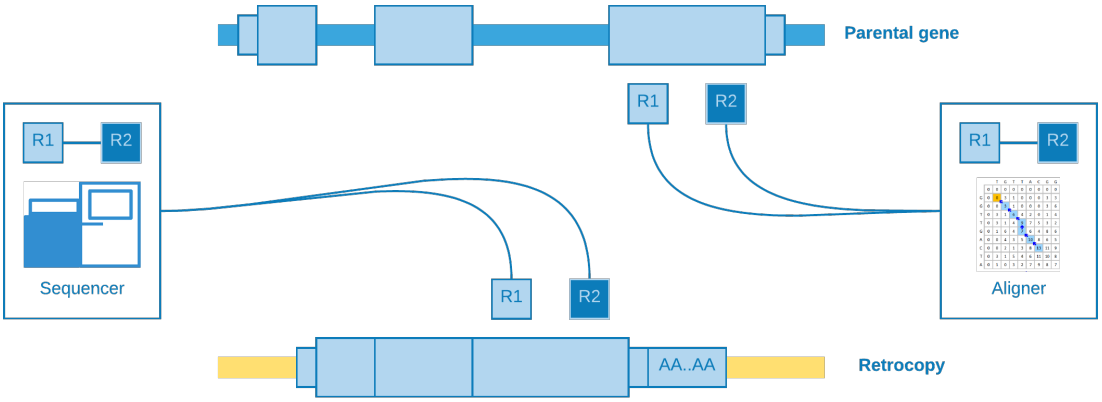
Methodology

The sideRETRO **methodology** consists of detecting abnormal (discordant) alignments in SAM/BAM file and, with an **unsupervised machine learning** algorithm, clustering these reads and genotype in order to discover somatic retrocopy insertions. Care is taken to ensure the quality and consistency of the data, taking into account the features that characterize a retrocopy mobilization, such as the absence of **intronic** and **regulatory** regions.

Note: For more detail about the jargon, see Retrocopy in a nutshell

5.1 Abnormal alignment

When a structural variation, such as a retrotransposition, occurs into an individual and her genome is sequenced with a next-generation sequencing technology (e.g. illumina), we may **expect** that the aligner (e.g. BWA, Bowtie) will be **confused** as to the origin of certain reads. As the retrocopies come from a **mature mRNA**, reads from the retrocopy may be **erroneously** aligned to an **exon** of the **parental gene**:



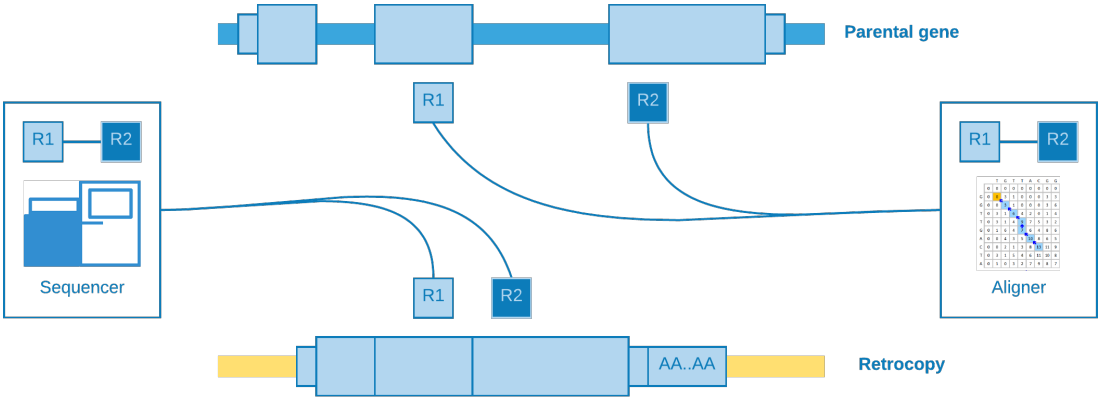
These kind of alignment may be called **indistinguishable**, because they do not give any **clue** about the presence of the retrocopy. However, for our luck, there are reads with abnormal (discordant) alignments which could be helpful according to their characteristics:

- Paired-end reads aligned at **different** exons
- Paired-end reads aligned at **different** chromosomes
- Paired-end reads aligned at **distant** regions
- Splitted reads (Reads with **supplementary** alignment)

We will talk about each one as best as we can in the next lines.

5.1.1 Alignment at different exons

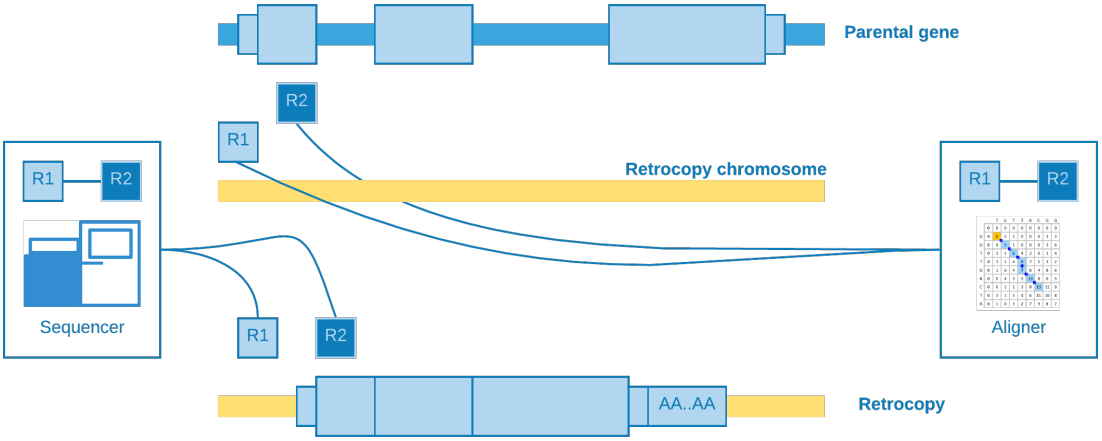
When paired-end reads are mapped to contiguous exons and they came from a genomic sequencing - which of course is not expected.



This kind of alignment is useful for **assume** a retrotransposition for the given parental gene, however it is not possible to annotate the **genomic position** of the event.

5.1.2 Alignment at different chromosomes

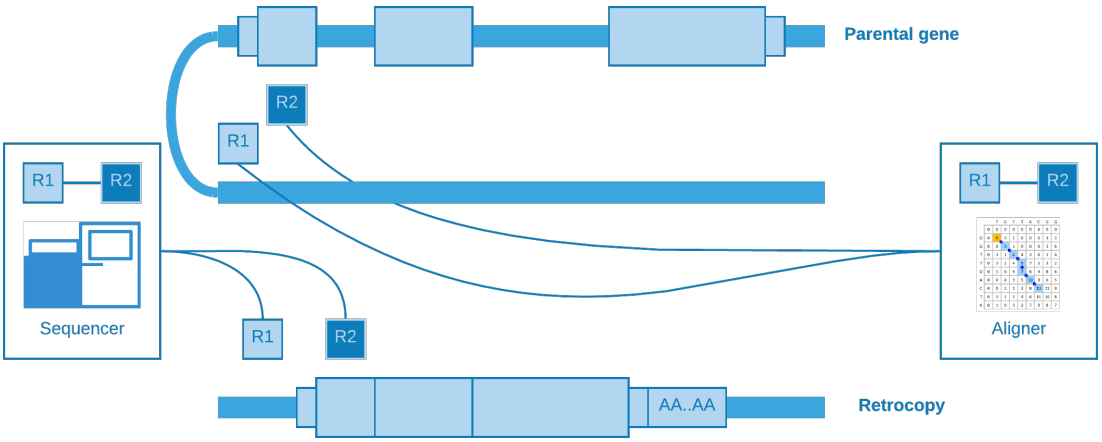
When the retrotransposition does not occur into the **same** parental gene chromosome, it may happen that one read come from a **near** intergenic region and its pair from the somatic **retrocopy**. As the retrocopy **does not exist** in the reference genome, the aligner will **map** one read to the retrotransposition chromosome and its pair to the parental gene **exon**.



This alignment is useful to **estimate** the genomic position of the event, but not with so much **precision** concerning to the **insertion point**.

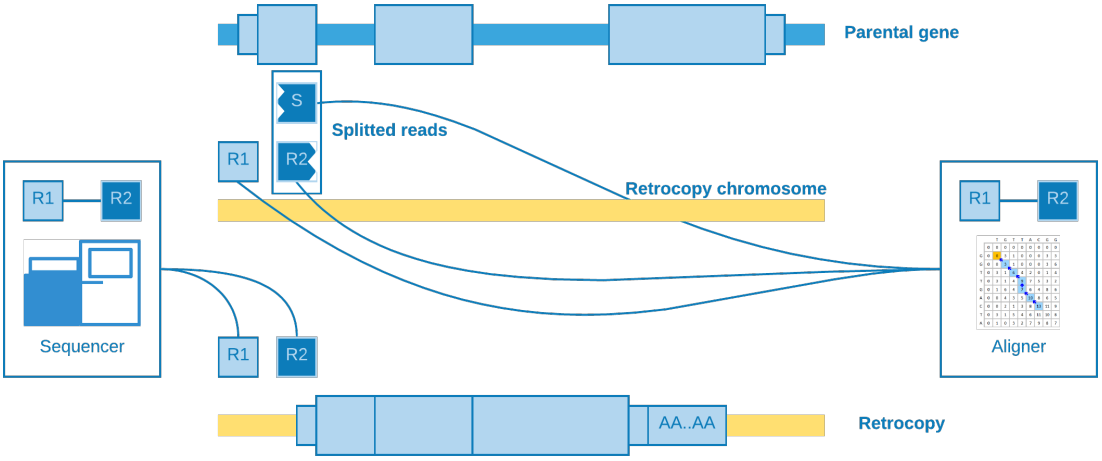
5.1.3 Alignment at distant regions

If a retrocopy is inserted into the **same** chromosome of its parental gene, possibly it will occur at a **distant** location. As well as *"alignment at different chromosomes"*, one read may come from a near intergenic region and its mate from the somatic retrocopy. So when the aligner try to map these reads, we will observe that one fall inside the parental gene exon, while its pair is mapped to a **distant region**.



5.1.4 Splitted reads

The most **important** kind of alignment when detecting structural variations. The splitted read may occur when **part** of the **same** read come from a near intergenic region and part from the somatic retrocopy. When the aligner **try** to map the read, it will need to **create** another one to represent the splitted part, which is called **supplementary**.



This alignment is useful to detect the **insertion point** with a **good precision**.

5.1.5 Taking all together

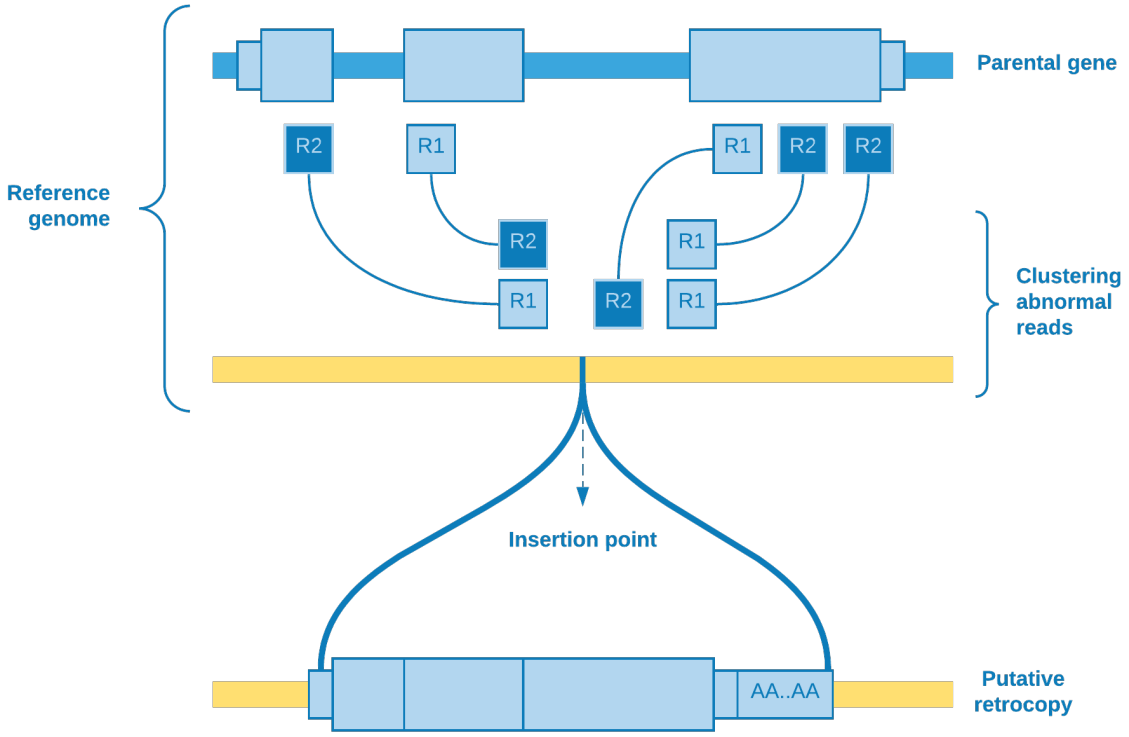
We can resume all abnormal alignments according to their power to detect the retrotransposition coordinate and its exact insertion point:

Abnormal alignments	Coordinate	Insertion point
At different exons	NO	NO
At different chromosomes	YES	NO
At distant regions	YES	NO
Splitted read	YES	YES

sideRETRO uses **only** the abnormal alignments **capable** to detect **at least** the coordinate, so those that fall into *different exons* are dismissed.

5.2 Clustering

So far we have been talking about abnormal reads **selection**. As soon as this step is over, we need to determine if a bunch of reads aligned to some genomic region may **represent** a putative retrocopy insertion. Therefore, firstly we restrict the abnormal reads for those whose **mate is mapped** to a protein coding **exon**, and then we **cluster** them according to the chromosome they mapped to.

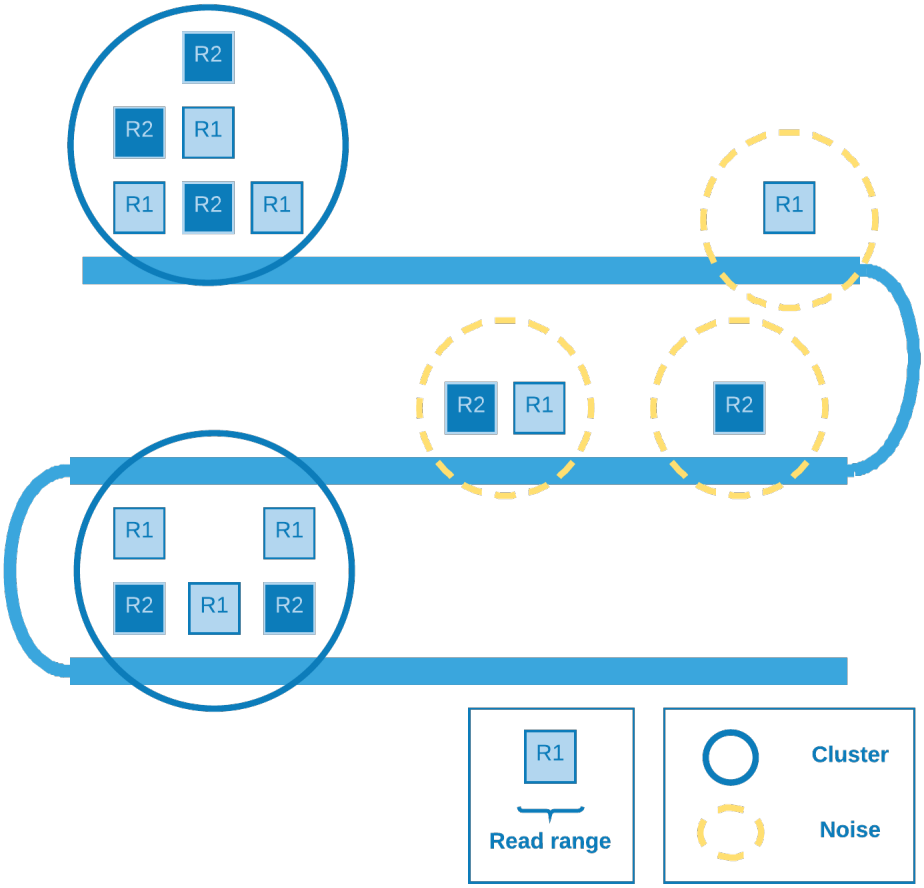


Wherefore, the clustering algorithm plays the role to resolve if there really is a retrotransposition event. As the **number** of reads **covering** the group is an important feature to take into account, one possible choice of algorithm is **DBSCAN**.

5.2.1 DBSCAN

Density Based Spatial Clustering of Applications with Noise[1] is a density based clustering algorithm designed to discover cluster in a **spatial database**. In our particular case, the database is spatially of **one dimension** (the chromosome extension) and the points are represented by the **range** comprising the mapped reads start and end.

[1] Ester, Martin. (1996). A Density-Based Algorithm for Discovering Clustersin Large Spatial Databases with Noise. KDD. Available at https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf.



The denser (covered) the region the greater the chance of a retrotransposition event there.

5.3 Genotype

In order to **increase** the putative insertion coverage, it is common to **join** analysis of a bunch of individuals. After the discovery of the retrocopies, it is necessary to identify **who owns** the variation and with what **zygosity** ((heterozygous, homozygous). So we have **three** possibilities for biallelic sites[2]: If *A* is the **reference** allele and *B* is the **alternate** allele, the ordering of genotypes for the likelihoods is *AA*, *AB*, *BB*. The **likelihoods** in turn is calculated based on *Heng Li* paper[3] with some assumptions that we are going to discuss.

Suppose at a given retrotransposition insertion point site there are *k* reads. Let the first *l* reads identical to the reference genome and the rest be different. The unphred alignment error probability of the *j*-th read is ϵ_j. Assuming error independence, we can derive that:

$$\delta(g) = \frac{1}{m^k} \prod_{j=1}^{l} [(m-g)\epsilon_j + g(1-\epsilon_j)] \prod_{j=l+1}^{k} [(m-g)(1-\epsilon_j) + g\epsilon_j]$$

where:

[2] hts-specs. (2019). The Variant Call Format (VCF) Version 4.2 Specificatio. Available at https://samtools.github.io/hts-specs/VCFv4.2.pdf.

[3] Li, Heng (2011). A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. Oxford University Press.

$\delta(g)$	Likelihoods for a given genotype
m	Ploidy
g	Genotype (the number of reference alleles)

Note: The way we are modeling the likelihoods probability **differs** a little bit from the **SNP calling** model: We are **treating** the *read* as the **unit**, not the *base*, therefore the error (ϵ) is the **mapping** quality (fifth column of BAM file), instead of the **sequencing** quality.

So we can summarize the formula for homozygous reference (HOR), heterozygous (HET) and homozygous alternate (HOA):

HOR

$$\delta(HOR) = \frac{1}{2^k} \prod_{j=1}^{l} 2(1-\epsilon_j) \prod_{j=l+1}^{k} 2\epsilon_j$$

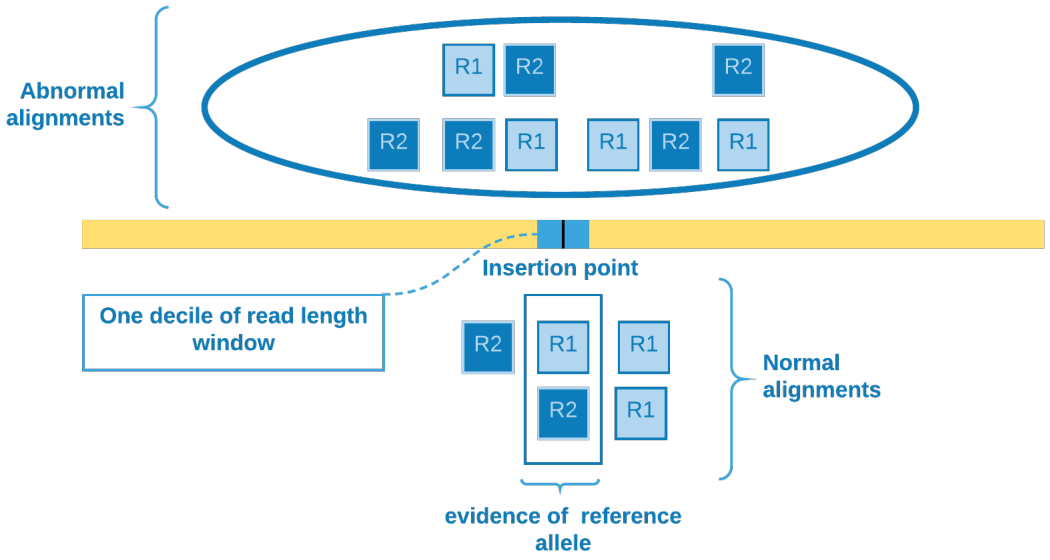
HET

$$\delta(HET) = \frac{1}{2^k}$$

HOA

$$\delta(HOA) = \frac{1}{2^k} \prod_{j=1}^{l} 2\epsilon_j \prod_{j=l+1}^{k} 2(1-\epsilon_j)$$

We determine the insertion point site according to the abnormal alignments clustering. Those *reads* will be used as the $k-l$ rest of the *reads* which differs from reference genome. In order to verify if there is evidence of reference allele, we need to come back to the BAM file and check for the presence of *reads* **crossing** the insertion point. To **mitigate** alignment error - which would otherwise overestimate the number of reference allele *reads* - we select the *reads* that cover one **decile** of *read* length window containing the insertion point. Then we come to the l *reads* **identical** to the reference genome and can calculate the **genotype likelihoods**.



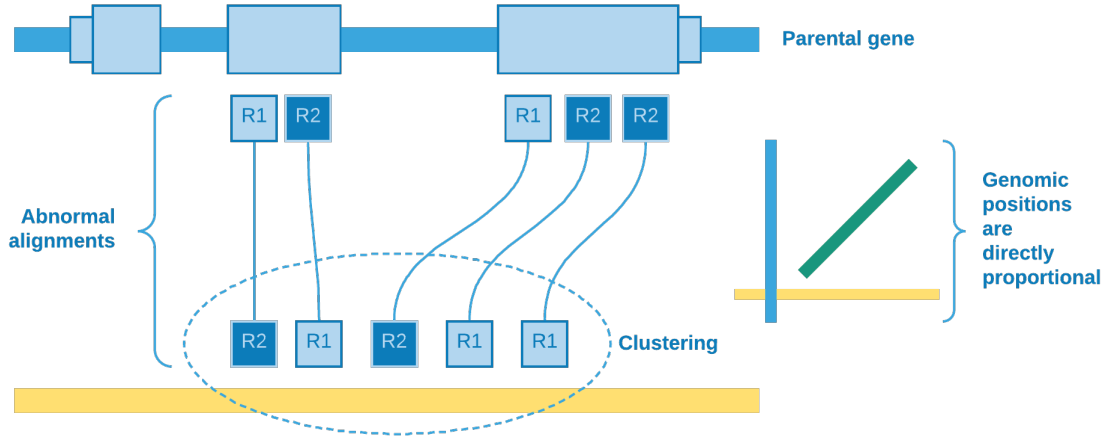
5.4 Orientation

Other important information that can be obtained from the data is the retrocopy **orientation** in relation to its parental gene. The abnormal alignment *reads* give us the clue to solve this issue. We catch *reads* when one pair aligns against an exon and its mate aligns to some genomic region, so we can **sort** the *reads* from the exonic site and analyze if their mates will be sorted in **ascending** or **descending** order as result. If we observe that they are **directly** proportional, then we can assume that the retrocopy is at the **same** parental gene strand, else they are at **opposite** strands.

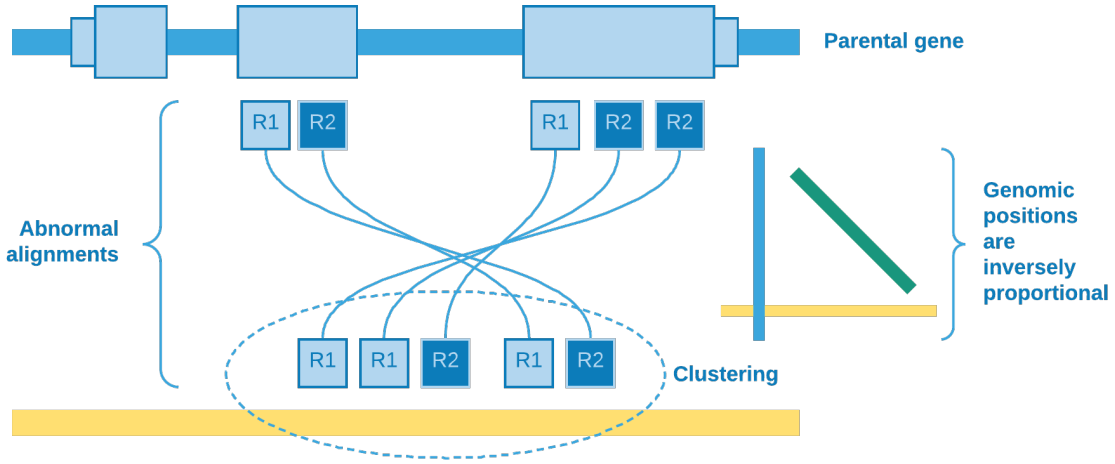
> **Warning:** This approach disregards the fact that there may have been structural variations, such as chromosomal inversions, which may invalidate these results.

Therefore summarizing:

- Retrocopy and its parental gene are at the same strand



- Retrocopy and its parental gene are at opposite strands



5.4.1 Spearman's rank correlation coefficient

We use *Spearman's rank correlation coefficient*[4] in order to have a **measure** of relationship between *reads* from exon and their mates from clustering site. As our data is **nonparametric**, the Spearman's rho can assess **monotonic** relationship, that is, it can tell us if the genomic position of *reads* from exon **increases** when **does** the genomic position of their *mates* from clustering site (positive rho) - or the opposite (negative rho).

So we come to the following proposition:

Parental gene strand	Retrocopy strand	
	rho > 0	rho < 0
+	+	-
-	-	+

5.5 References and Further Reading

[4] Fieller, E. C., et al. (1957). Tests for Rank Correlation Coefficients. I. Biometrika, 44(3/4), 470–481. JSTOR. Available at https://www.jstor.org/stable/2332878.

CHAPTER 6

Results

To assess **sideRETRO** performance and accuracy, we made a simulated dataset whereupon we could have control of **true** positive and negative values.

6.1 Dataset

Our dataset for testing is composed of 5 simulated human whole-genome sequencing with 40x of depth and 10 randomly distributed retrocopies each. In total, we had a list of 30 retrocopies consisting of the last 1000 bases of the largest transcript of the parental gene. All retrocopies was stochastically designed for chromosome, position, strand and zygosity.

Table 1: 30 parental gene names chosen for the simulation

ACAP3	AMY2B	ATM	BCL2	BRAF
BRCA1	BRIP1	DVL1	FANCD2	FAT1
GNB1	KLHL17	MECP2	MUTYH	NPHP4
OR4F5	PBX1	PRCC	PTEN	RER1
SAMD11	SET	SIRT1	SMARCE1	SOX2
TMEM52	TP53	TPM3	USP8	XPO1

For each individual human whole-genome, we **sampled with replacement** 10 retrocopies from our list, so that some events were shared among part of our subjects.

Table 2: All randomly designed retrocopies for simulation. *HE heterozygous and *HO homozygous alternate

Parental gene	Chromosome	Position	Strand	Zygosity
AMY2B	chr5	122895832	-	HE
ACAP3	chr11	133246346	+	H0
ATM	chr19	16443740	+	HO
BCL2	chr4	146453786	+	HO
BRAF	chr18	22375812	-	HO
BRCA1	chr7	102911369	-	HO
BRIP1	chr12	113357216	-	HO
DVL1	chr5	54105196	-	HE
FANCD2	chr3	121339674	+	HE
FAT1	chr16	65659952	-	HE
GNB1	chr12	70734022	+	HE
MUTYH	chr4	2716745	+	HO
PBX1	chr21	22718521	-	HE
PRCC	chr1	190777903	-	HO
PTEN	chr2	140252560	-	HE
RER1	chr13	55179109	+	HE
SAMD11	chr4	14585135	+	HO
SET	chr7	154178578	-	HO
SIRT1	chrX	120716688	-	HO
SOX2	chr10	88689163	-	HE
TMEM52	chr1	82897536	+	HO
TP53	chr5	42938944	-	HO
TPM3	chr7	3488208	-	HE
USP8	chr7	41333317	+	HO
XPO1	chr10	33172062	-	HE

Table 3: Parental gene name for the retrocopies sampled by individual whole-genome

IND1	IND2	IND3	IND4	IND5
ATM	AMY2B	ATM	AMY2B	AMY2B
BCL2	BRCA1	GNB1	BRAF	BRIP1
BRIP	FANCD2	MECP	BRIP1	FANCD2
DVL1	FAT1	PBX1	DVL1	NPHP4
MECP	KLHL17	PRCC	FANCD2	PTEN
OR4F	MUTYH	PTEN	GNB1	RER1
PTEN	RER1	RER1	KLHL17	SMARCE1
SET	SET	SAMD	MECP2	TP53
SOX2	SMARCE1	SIRT	SET	TPM3
XPO1	SOX2	XPO1	TMEM52	XPO1

6.2 Simulation

We used the **SANDY** tool (version v0.23), *A straightforward and complete next-generation sequencing read simulator*[1], for simulate all 5 genomes according to the structural variations that we designed and according to the sampling.

[1] Miller, Thiago et al. (2019). galantelab/sandy: Release v0.23 (Version v0.23). Zenodo. http://doi.org/10.5281/zenodo.2589575.

SANDY demands 2 steps for the task: First we indexed all retrocopies for each individual and then we could simulate all whole-genome sequencing.

```
# Reference genome
REF_FASTA=hg38.fa

# Coverage depth
DEPTH=40

# Retrocopies by individual
IND=(ind1.txt ind2.txt ind3.txt ind4.txt ind5.txt)

# Index all retrocopies
for ind in "${IND[@]}"; do
  sandy variation add --structural-variation=${ind%%.txt} $ind
done

# Simulate all genomes
for ind in "${IND[@]}"; do
  sandy_index=${ind%%.txt}
  sandy genome \
    --id='%i.%U_%c:%S-%E_%v' \
    --structural-variation=$sandy_index \
    --output-dir=$sandy_index \
    --jobs=20 \
    --seed=42 \
    --quality-profile='hiseq_101' \
    --coverage=$DEPTH \
    --verbose \
    $REF_FASTA
done
```

As result we have a pair of FASTQ files (forward and reverse complement) for each simulated individual. Next it is required to align our sequencing data against the human reference genome in order to generate mapped files in SAM format. We used BWA aligner (version 0.7.9)[2] for accomplish this task.

```
# Individual directories with the
# simulated data
IND_DIR=(ind1 ind2 ind3 ind4 ind5)

# Reference genome
REF_FASTA=hg38.fa

# Index reference genome
bwa index $REF_FASTA

# Alignment
for ind in "${IND[@]}"; do
  bwa mem \
    -t 10 \
    $REF_FASTA \
    $ind/out_R1_001.fastq.gz \
    $ind/out_R2_001.fastq.gz > $ind/out.sam
done
```

[2] Li H. and Durbin R. (2009). Fast and accurate short read alignment with Burrows-Wheeler Transform. Bioinformatics, 25:1754-60. [PMID: 19451168].

6.3 Running sideRETRO

After our simulated dataset was ready, we could test the sideRETRO capabilities to detect the designed somatic retrocopies.

```
# Our simulated SAM files list
LIST=(ind1/out.sam ind2/out.sam ind3/out.sam ind4/out.sam ind5/out.sam)

# GENCODE annotation v31
ANNOTATION=gencode.v31.annotation.gff3.gz

# GENCODE reference genome
REF_FASTA=hg38.fa

# Run process-sample step
sider process-sample \
  --cache-size=20000000 \
  --output-dir=result \
  --threads=5 \
  --max-distance=15000 \
  --alignment-frac=0.9 \
  --phred-quality=20 \
  --sorted \
  --log-file=ps.log \
  --annotation-file=$ANNOTATION \
  "${LIST[@]}"

# Run merge-call step
sider merge-call \
  --cache-size=20000000 \
  --epsilon=500 \
  --min-pts=20 \
  --genotype-support=5 \
  --near-gene-rank=3 \
  --log-file=mc.log \
  --threads=10 \
  --phred-quality=20 \
  --in-place \
  result/out.db

# Finally run make-vcf
sider make-vcf \
  --reference-file=$REF_FASTA \
  --prefix=simulation \
  result/out.db
```

6.4 Download

Our output `simulation.vcf`, as well as the files to be indexed by SANDY, `ind1.txt`, `ind2.txt`, `ind3.txt`, `ind4.txt` and `ind5.txt`, can be downloaded at `simulation.tar.gz`.

6.5 Analysis

All retrocopies were detected except for 5 retrocopies from the following parental genes that were not detected in any individual: `KLHL17`, `MECP2`, `NPHP4`, `OR4F5` and `SMARCE1`. In a manual check, those retrocopies do not present *abnormal reads* in any individual - which could enable their recognition by our pipeline. However to make a fair and reproducible analysis, we will consider it as a methodological error of our tool. So sideRETRO found $\frac{23}{28}$ retrocopies, which means **82%**.

In more detail, we will show that result and more concerning:

1) *Genomic coordinate*

2) *Zygosity*

6.5.1 Genomic coordinate

Regarding the genomic coordinate, sideRETRO got it right **100%** for **chromosome** and **strand**. The performance in detecting the insertion point position was satisfactory with a MEDSE of **1 base**.

Table 4: Predicted position. Error = $|actualPos - predictedPos|$

Parental gene	Actual position			Predicted position			Error
AMY2B	chr5	122895832	-	chr5	122895831	-	1
ATM	chr19	16443740	+	chr19	16443738	+	2
BCL2	chr4	146453786	+	chr4	146453783	+	3
BRAF	chr18	22375812	-	chr18	22375811	-	1
BRCA1	chr7	102911369	-	chr7	102911368	-	1
BRIP1	chr12	113357216	-	chr12	113357216	-	0
DVL1	chr5	54105196	-	chr5	54105195	-	1
FANCD2	chr3	121339674	+	chr3	121339673	+	1
FAT1	chr16	65659952	-	chr16	65659951	-	1
GNB1	chr12	70734022	+	chr12	70734022	+	0
MUTYH	chr4	2716745	+	chr4	2716760	+	15
PBX1	chr21	22718521	-	chr21	22718520	-	1
PRCC	chr1	190777903	-	chr1	190777902	-	1
PTEN	chr2	140252560	-	chr2	140252559	-	1
RER1	chr13	55179109	+	chr13	55179108	+	1
SAMD11	chr4	14585135	+	chr4	14585134	+	1
SET	chr7	154178578	-	chr7	154178578	-	0
SIRT1	chrX	120716688	-	chrX	120716687	-	1
SOX2	chr10	88689163	-	chr10	88689162	-	1
TMEM52	chr1	82897536	+	chr1	82897534	+	2
TP53	chr5	42938944	-	chr5	42938943	-	1
TPM3	chr7	3488208	-	chr7	3488207	-	1
XPO1	chr10	33172062	-	chr10	33172056	-	6

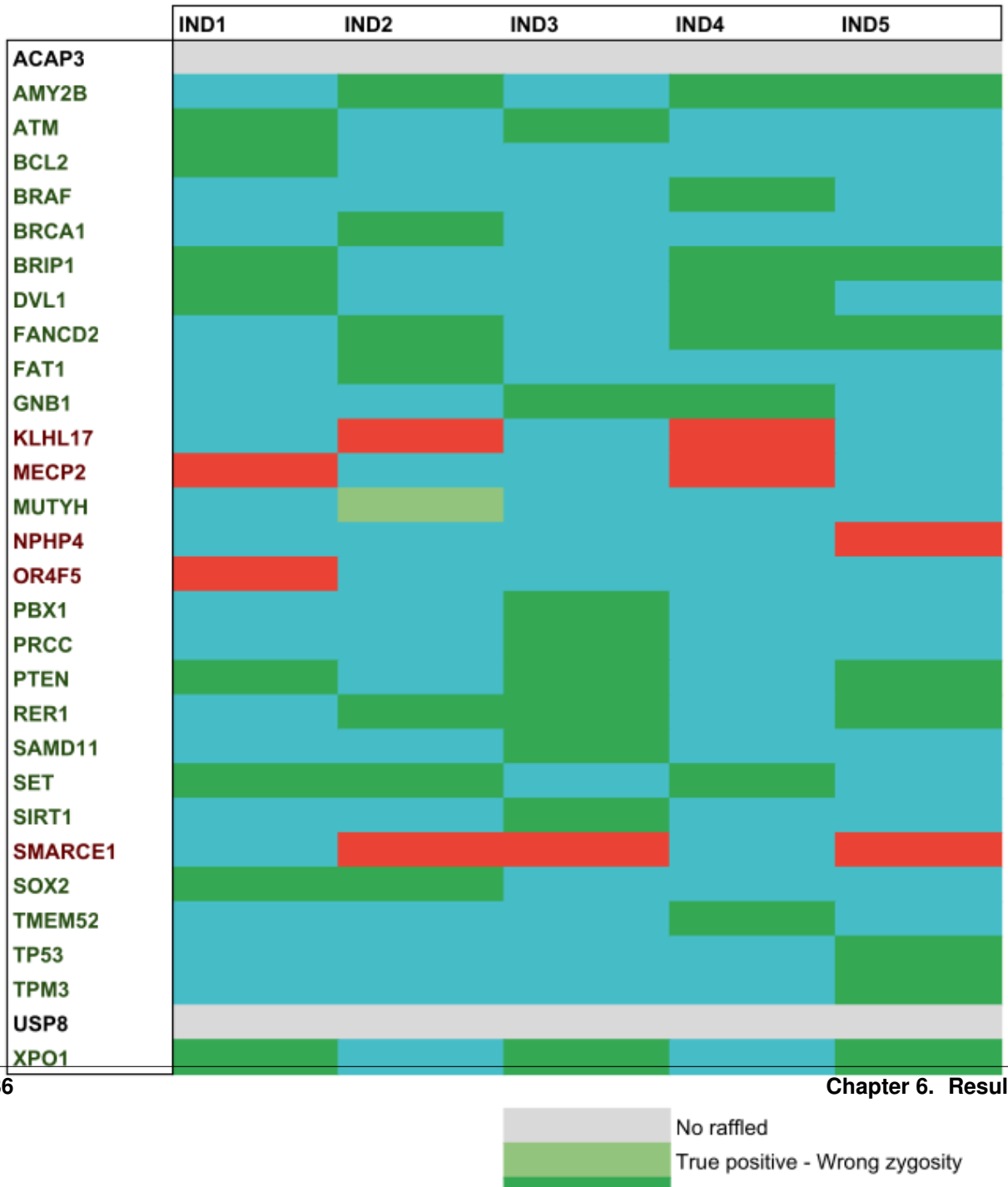
Table 5: Genomic coordinate detection summary. MSE (Mean Squared Error), MEDSE (Median Squared Error).

Chromosome	Strand	Position	
		MSE	MEDSE
100%	100%	12.7 bases	1 base

Note: The retrocopies from `MUTYH` and `XPO1` do not present *splitted reads*, so they are annotated as *IMPRECISE* at `simulation.vcf` file. So it can explain why their insertion point position detection has the biggest absolute error, which pulls the MSE up.

6.5.2 Zygosity

With the exception of the 5 undetected retrocopies (parental genes `KLHL17`, `MECP2`, `NPHP4`, `OR4F5` and `SMARCE1`), it was possible to accurately calculate the zygosity. Only for the `MUTYH`'s retrocopy, the zygosity was wrongly assigned as heterozygous for the individual `IND2` - when the true value was *homozygous alternate*. As `MUTYH`'s retrocopy was *imprecisely* annotated for the lack of splitted reads, we may expect a reference allele depth overestimation on the predicted site - which explains the heterozygous attribution.



To calculate the performance in terms of reference/alternate allele detection, we made a confusion matrix for each individual and an overall matrix

with the junction of all simulations:

There was no *false positive* at all - as can be seen by the **precision** and **specificity** with 100% value for all individuals. The worst **accuracy** was at individual 2, with 91.07% value, and best was at individual 3, with 96.43% value. The overall **sensitivity**, **specificity** and **accuracy** were: 81%, 100% and 93.21% value consecutively.

6.6 References and Further Reading

CHAPTER 7

Authors

- Thiago Luiz Araujo Miller <tmiller@mochsl.org.br>
- José Leonel Lemos Buzzo <lbuzzo@mochsl.org.br>
- Fernanda Orpinelli <forpinelli@mochsl.org.br>
- Pedro Alexandre Favoretto Galante <pgalante@mochsl.org.br>

- search