sideRETRO Documentation

Thiago L. A. Miller

Jul 01, 2023

Contents:

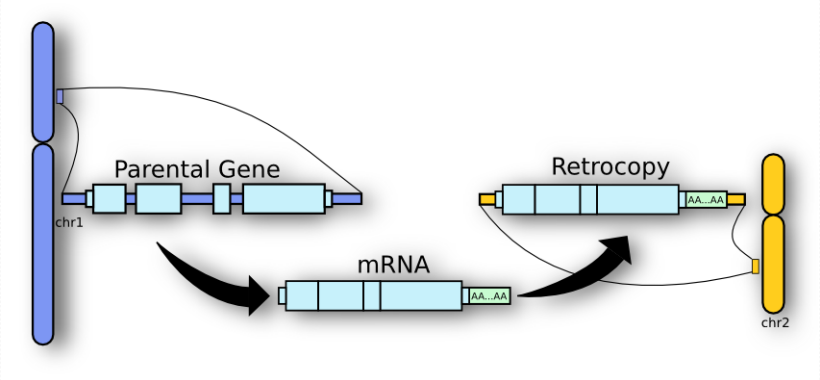
CHAPTER 1

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/CRAM 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**.

Strandness 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/CRAM 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 Citation

If sideRETRO was somehow useful in your research, please cite it:

```
@article{10.1093/bioinformatics/btaa689,
  author = {Miller, Thiago L A and Orpinelli, Fernanda and Buzzo, José Leonel L and␣
↪Galante, Pedro A F},
  title = "{sideRETRO: a pipeline for identifying somatic and polymorphic insertions␣
↪of processed pseudogenes or retrocopies}",
  journal = {Bioinformatics},
  year = {2020},
  month = {07},
  issn = {1367-4803},
  doi = {10.1093/bioinformatics/btaa689},
  url = {https://doi.org/10.1093/bioinformatics/btaa689},
  note = {btaa689},
}
```

1.8 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`:

```
$ sudo apt install meson
```

Or "$ pip3 install –user meson" But in this case, remember to set the environment variables.

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
   -c, --cite            Show citation in BibTeX

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 1.0.0
```

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. sider will look for 'exon' with the attribute 'transcript_type=protein_coding'. The attributes 'gene_name', 'gene_id' and 'exon_id' are also required
-i, --input-file	File containing a newline separated list of alignment files in SAM/BAM/CRAM format. This option is not mandatory if one or more SAM/BAM/CRAM 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/CRAM 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 (SAM/BAM/CRAM) 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 Dealing with CRAM format

Working with CRAM files may be a little **tricky**, mainly if you have downloaded the data from a public repository. Let's take a look at two possible cases:

- Local alignment
- External alignment

3.5.1 Local alignment

In order to generate an alignment file in the CRAM format, first we need to index the reference genome:

```
# Inde for BWA: .fa.amb, .fa.ann, .fa.bwt, .fa.pac, .fa.sa files
bwa index hg38.fa

# Index reference genome for CRAM: .fa.fai file
samtools faidx hg38.fa
```

Then, we can align with `bwa`:

```
# Align with BWA and generate a CRAM
bwa mem hg38.fa file_R1.fastq file_R2.fastq | \
   samtools view -T hg38.fa -C -o file.cram -
```

The alignment `file.cram` can be processed with `sider`, as long as we don't change the reference genome and its index (`.fa.fai`) path. If so, we need to set the environment variables `REF_PATH` and `REF_CACHE`, see *External alignment*.

3.5.2 External alignment

When we download public data already aligned in the CRAM format, we may be concerned about the reference genome index. Probably, we won't have the required genome index to read the `.cram`, and the `htslib` library - used by `sider` and `samtools` - is able to download the index from the CRAM Reference Registry.

However, in order to `htslib` be able to accomplish this task, we need to compile the library with the required flags and also we need to have the reqeuired dependencies (as libcurl). Therefore to be able to read these files, without depending on these details, we need to generate a new local index and set the environment variables - `REF_PATH` and `REF_CACHE` - to the correct path:

```
# Create cache dir
mkdir -p /my/cache

# Construct the index
perl seq_cache_populate.pl -root /my/cache hg38.fa

# Now before running samtools or sider, we need to
# set the environment variables REF_PATH and REF_CACHE
export REF_PATH=/my/cache/%2s/%2s/%s:http://www.ebi.ac.uk/ena/cram
export REF_CACHE=/my/cache/%2s/%2s/%s

# So ...
sider ps -a annot.gff3.gz -o result file.cram
```

The script `seq_cache_populate.pl` can be found in the `samtools`, or at seq_cache_populate.pl.

For more information, see Samtools Worflow.

3.6 A Practical Workflow

Now, let's do an interesting exercise, with real experimental data from the 1000 Genomes Project. (Warning: This example requires 16GB of RAM)

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.

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 FASTA file with the gencode's Human reference genome, version 38 (here `GRCh38_full_analysis_set_plus_decoy_hla.fa`).
3. A custom perl script, `seq_cache_populate.pl`, to construct a new local index . The `seq_cache_populate.pl` script can be found in seq_cache_populate.pl.
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`.

Also, we will set the environment variables `REF_PATH` and `REF_CACHE`, as a requirement to work with CRAM files - more information at *Dealing with CRAM format*.

See the complete command sequence below for the whole analysis.

Tip: Copy and paste line by line in your terminal.

Tip 2: If you are running line by line in your terminal don't paste the "$" character. It is already in your terminal.

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

(continues on next page)

(continued from previous page)

```
# Download annotation from gencode
wget ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_32/gencode.v32.
↪annotation.gtf.gz

# Download the reference genome from 1000 genomes
wget ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/GRCh38_reference_
↪genome/GRCh38_full_analysis_set_plus_decoy_hla.fa

# Make the CRAM index
# Create cache dir
mkdir -p cache

# create index
perl seq_cache_populate.pl -root cache GRCh38_full_analysis_set_plus_decoy_hla.fa

# Set environment variables
export REF_PATH=$PWD/cache/%2s/%2s/%s:http://www.ebi.ac.uk/ena/cram
export REF_CACHE=$PWD/cache/%2s/%2s/%s

# 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

# Create the list of BAM files.
# Average time: irrelevant
$ ls *.cram > 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.gz \
    -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 \
    -I \
    -t 2
```

(continues on next page)

(continued from previous page)

```
# 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 GRCh38_full_analysis_set_plus_decoy_hla.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.

3.7 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.

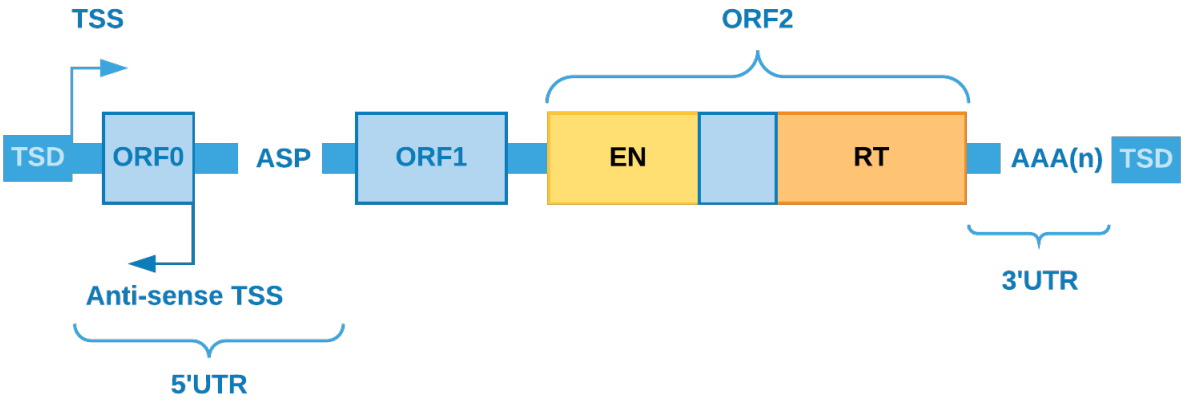
[3] DE KONING, A. P. J. et al. (2011). Repetitive Elements May Comprise Over Two-Thirds of the Human Genome. PLoS genetics, 7(12), e1002384.

[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

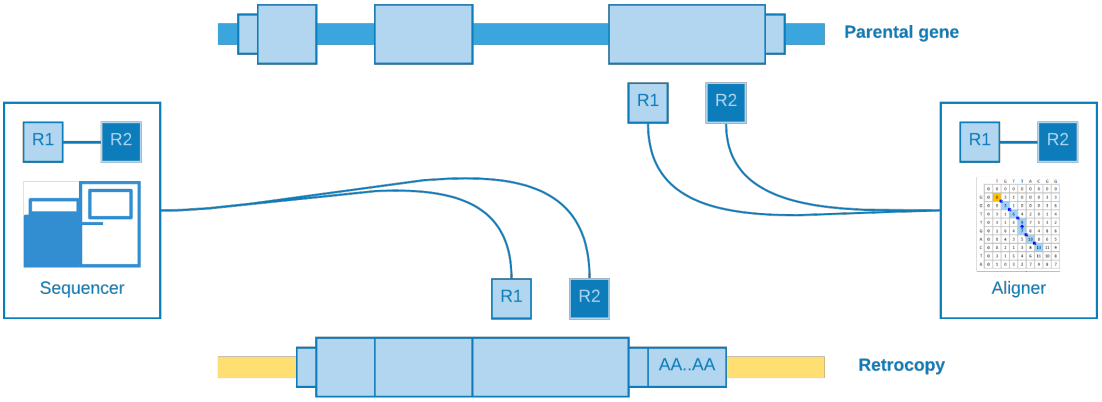
siderRETRO uses NGS (*Next Generation Sequencing*) data to identify **unfixed** - *dimorphic/polymorphic, germinative, or somatic* - retrocopies absent in the reference genome, but present in the sequenced genome (by NGS).

Our **methodology** consists of detecting abnormal (discordant) alignments in SAM/BAM/CRAM 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**:



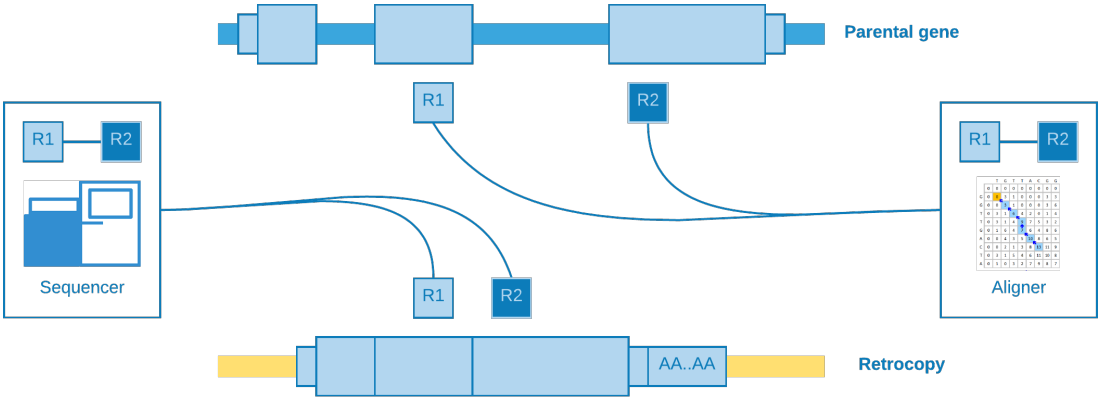
These kind of alignment may be called **indistinguishable**, because they do not give any **clue** about the presence of the retrocopy. However, for our luck, there are reads with abnormal (discordant) alignments which could be helpful according to their characteristics:

- 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

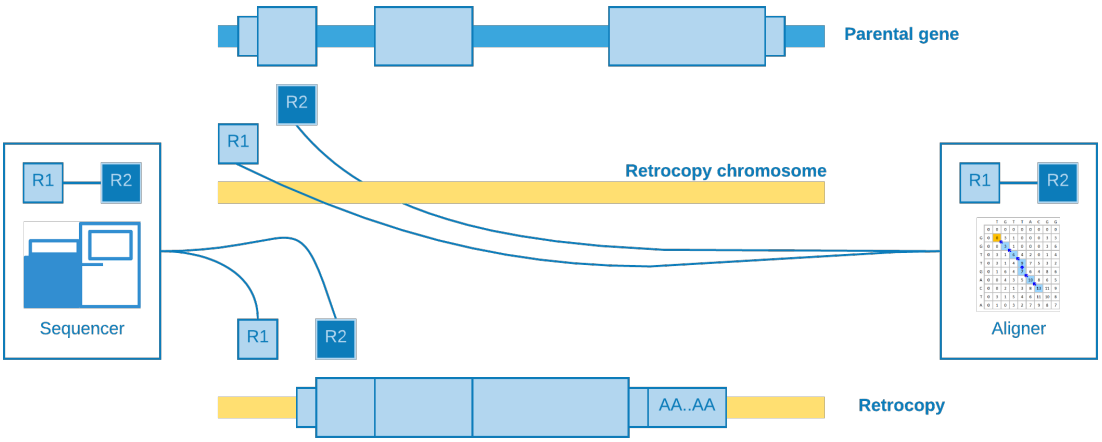
When paired-end reads are mapped to contiguous exons and they came from a genomic sequencing - which of course is not expected.



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

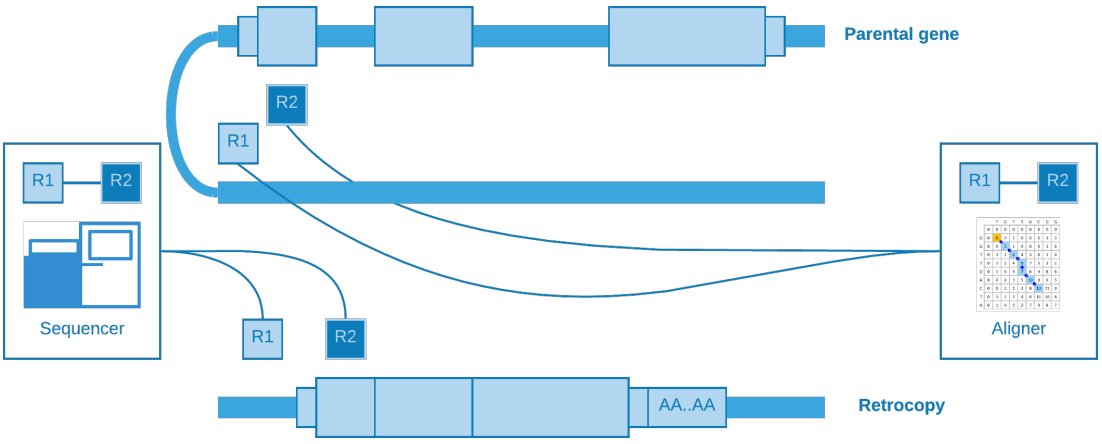
When the retrotransposition does not occur into the **same** parental gene chromosome, it may happen that one read come from a **near** intergenic region and its pair from the somatic **retrocopy**. As the retrocopy **does not exist** in the reference genome, the aligner will **map** one read to the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

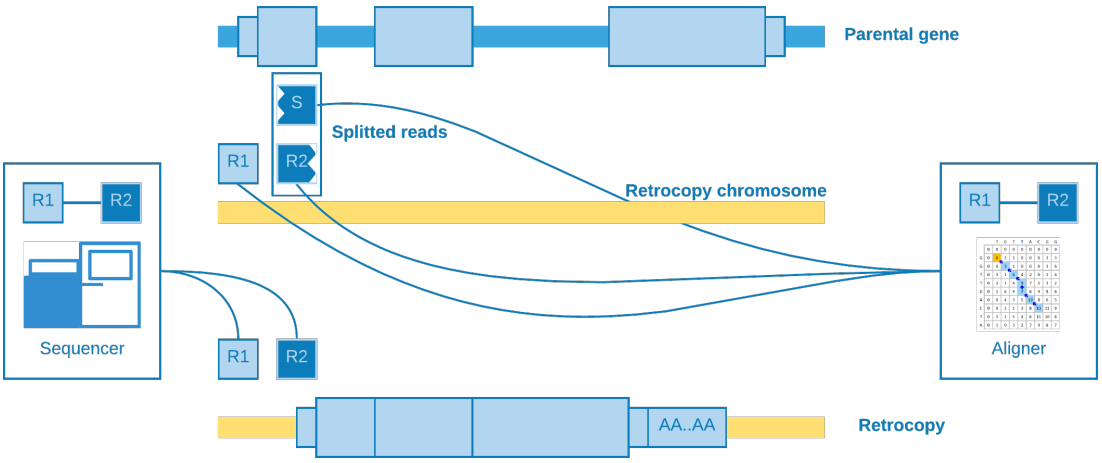
5.1.3 Alignment at distant regions

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

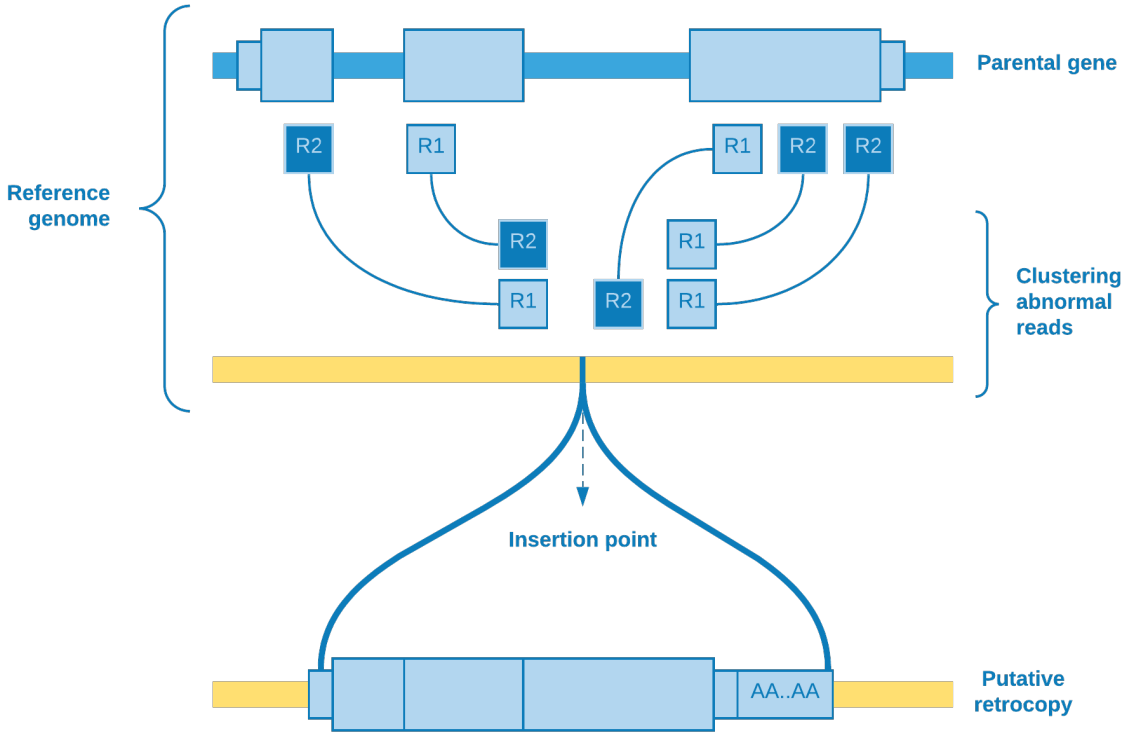
We can resume all abnormal alignments according to their power to detect the retrotransposition coordinate and its exact insertion point:

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.

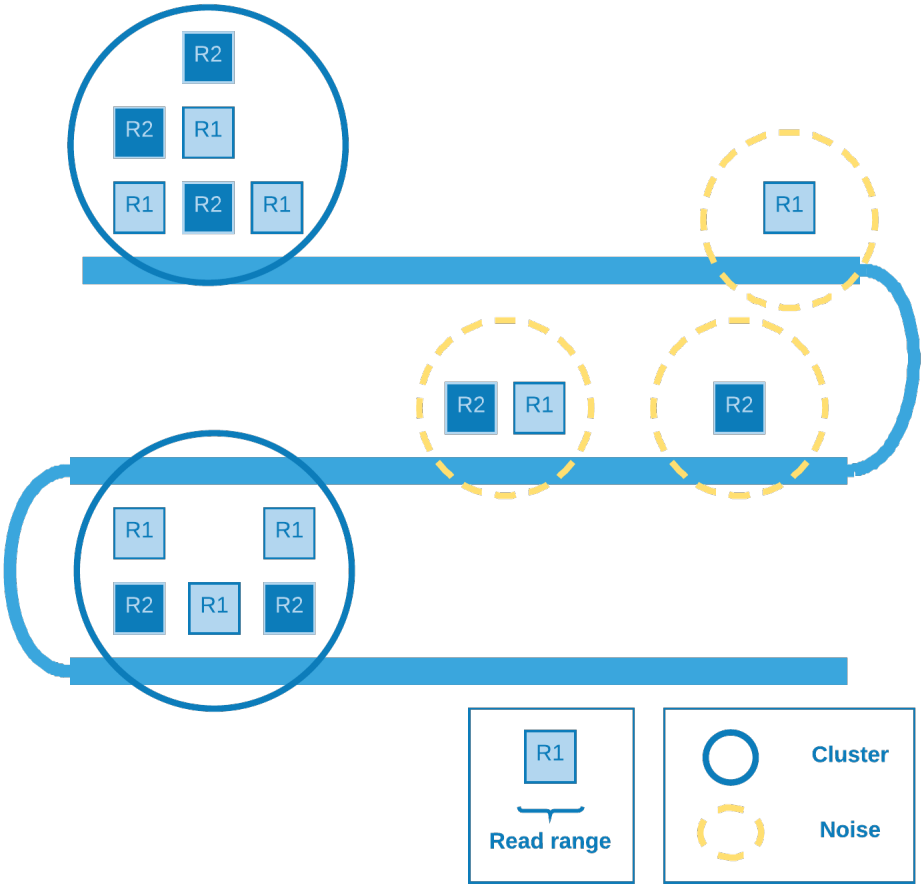


Wherefore, the clustering algorithm plays the role to resolve if there really is a retrotransposition event. As the **number** of reads **covering** the group is an important feature to take into account, one possible choice of algorithm is **DBSCAN**.

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 SAM file), instead of the **sequencing** quality.

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

HOR

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

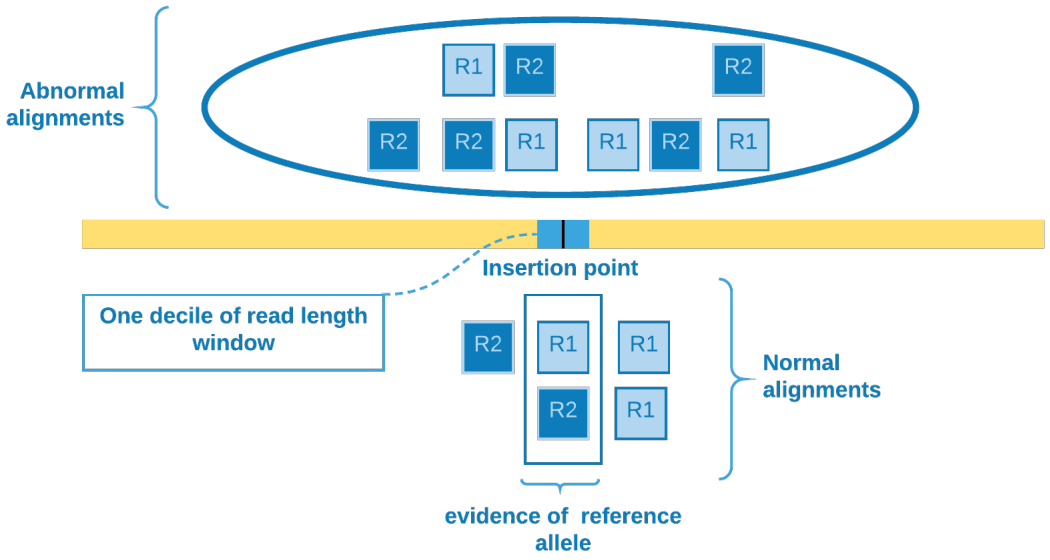
HET

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

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 SAM 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**.



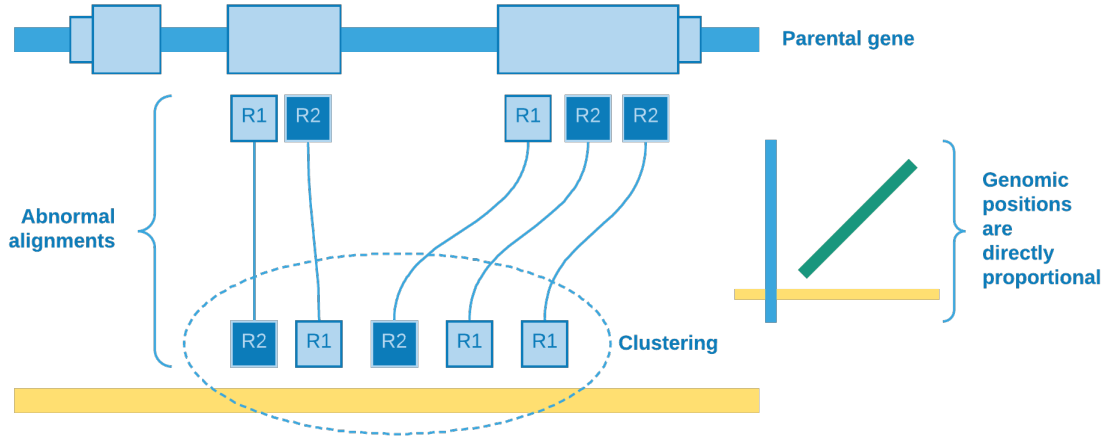
5.4 Orientation

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.

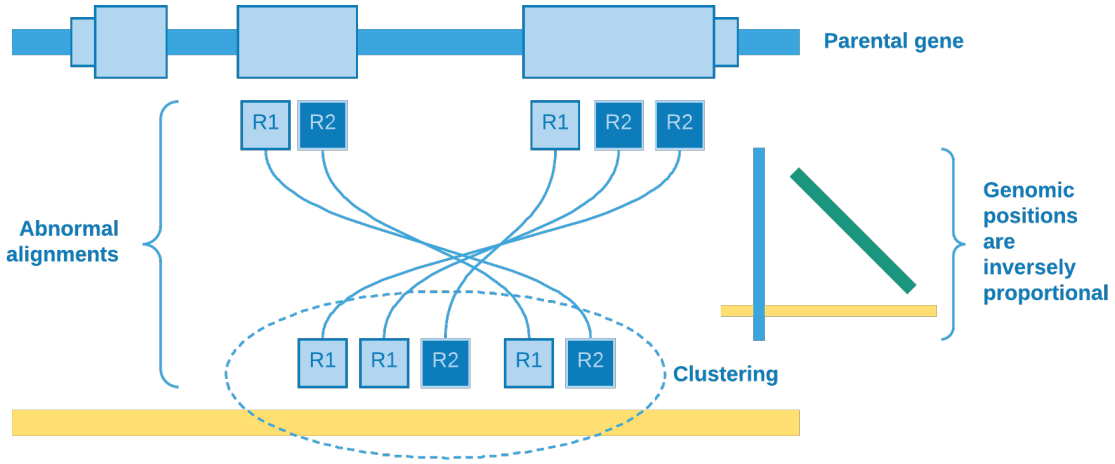
> **Warning:** This approach disregards the fact that there may have been structural variations, such as chromosomal inversions, which may invalidate these results.

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

Here are the results for simulated and real data.

Note: **retroCNVs** - *polymorphic retrocopies*

6.1 Simulated data

Our dataset for testing is composed of 100 simulated human whole-genome sequencing with 20x of depth and in average 30 randomly distributed retrocopies each. Simulation with low coverage of ('only') 20x in sequencing depth (i.e., heterozygotic events have only 10x coverage). This strategy allowed us to check the capability of sideRETRO to identify retroCNVs events even in a "non-ideal scenario" of low sequencing coverage. In total, we had a list of 100 retrocopies consisting of the last 1000 bases of the largest transcript of the parental gene - which were randomly raffled as well. All retrocopies was stochastically designed for chromosome, position, strand and zygosity.

The simulated retrocopy data is composed of three sets of retroCNVs events:

i) fixed or highly frequent events;

ii) polymorphic events (shared by some of the simulated genomes);

iii) somatic events (in only one genome) in simulation. It allowed us to check sideRETRO performance for these different types of retroCNVs.

6.1.1 Simulation

We developed a pipeline, which randomly generates our simulated dataset and make some analysis of performance. All scripts can be downloaded at `simulation.tar.gz`. We used the **SANDY** tool (version v0.23), *A straightforward and complete next-generation sequencing read simulator*[2], for simulate all 100 genomes according to the structural variations that we designed and according to the sampling. We used the reference human genome v38 and the GENCODE annotation v32.

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

```
REF_FASTA=/assets/hg38.fa
PC_FASTA=/assets/gencode.v32.pc_transcripts.fa
COHORT=100
RTC_NUM=100
LEN=1000
DEPTH=20
SANDY_SEED=1
SEED=17

# Genearte sequences
scripts/catch \
  --seed=$SEED \
  --rtc_num=$RTC_NUM \
  --length=$LEN \
  "$PC_FASTA" > rtc_100.tsv

# Build our cohort
scripts/build \
  --cohort=$COHORT \
  --seed=$SEED \
  --output-dir=build \
  "$REF_FASTA" \
  rtc_100.tsv

# Retrocopies by individual
IND=($(ls build/*.sandy))

# Load build values to SANDY
for ind in "${IND[@]}"; do
  sandy variation add \
    --structural-variation=$(basename $ind '.sandy') \
    $ind
done

mkdir -p sim

# Simulate all genomes
for ind in "${IND[@]}"; do
  sandy_index=$(basename $ind '.sandy')
  sandy genome \
    --id='%i.%U_%c:%S-%E_%v' \
    --structural-variation=$sandy_index \
    --output-dir="sim/$sandy_index" \
    --jobs=20 \
    --seed=$SANDY_SEED \
    --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)[3] for accomplish this task.

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

```
# Individual directories with the
# simulated data
IND_DIR=($(ls -d sim/*))

# Reference genome
REF_FASTA="/assets/hg38.fa"

# Index reference genome
bwa index $REF_FASTA

mkdir -p align

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

After our simulated dataset was ready, we run sideRETRO v0.14.1:

```
# Our simulated SAM files list
LIST=($(ls align/*.sam))

# GENCODE annotation v32
ANNOTATION=/assets/gencode.v32.annotation.gff3

# GENCODE reference genome
REF_FASTA=/assets/hg38.fa

# Run process-sample step
sider process-sample \
  --prefix=sim \
  --cache-size=20000000 \
  --output-dir=sider \
  --threads=20 \
  --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=10 \
  --log-file=mc.log \
  --threads=20 \
  --phred-quality=20 \
  --in-place \
  sider/sim.db
```

(continues on next page)

(continued from previous page)

```
# Finally run make-vcf
sider make-vcf \
  --log-file=vcf.log \
  --reference-file=$REF_FASTA \
  --prefix=sim \
  --output-dir=sider \
  sider/sim.db
```

Finally, with the sideRETRO's VCF made, we analysed the performance:

```
# Generate comparations for analysis
scripts/compare sider/sim.vcf build

# Confusion analysis
scripts/confusion analysis > confusion.tsv

# Just a look
$ column -t confusion.tsv | head
IND                 TP  FP  FN   PPV/Precision  TPR/Recall  F1-score
analysis/ind0.tsv   38  0   9    1.000000      0.808511    0.894118
analysis/ind1.tsv   36  2   11   0.947368      0.765957    0.847059
analysis/ind2.tsv   33  1   10   0.970588      0.767442    0.857143
analysis/ind3.tsv   35  1   12   0.972222      0.744681    0.843373
analysis/ind4.tsv   29  1   9    0.966667      0.763158    0.852941
analysis/ind5.tsv   37  4   12   0.902439      0.755102    0.822222
analysis/ind6.tsv   45  0   10   1.000000      0.818182    0.900000
analysis/ind7.tsv   37  2   11   0.948718      0.770833    0.850575
analysis/ind8.tsv   32  2   11   0.941176      0.744186    0.831169
```

6.1.2 Analysis

Table 1: Summary of the set of 100 simulated retroCNVs. lated retroCNV events were randomly inserted in the human g (GRCh38). Here, we present their parental gene name, the ins point, polarity (Pol). All events found (79 retroCNVs) and not (21 retroCNVs) are presented, as well as addition information abou insertion point (considering a region of 100bp around its position)

Parental	SIMULATED				FOUN
Gene	Chr	Position	Pol	LINE/SINE	Chr
ALG2	chr10	30778982	-	N	chr10
ARMC2	chr5	52723637	-	Y	chr5
ATG2B	chr5	177026995	-	N	chr5
BTF3	chr7	146774631	-	N	chr7
C2orf92	chr6	112158328	-	N	chr6
C8orf76	chr9	94927085	-	N	chr9
C9orf64	chr17	40139106	+	Y	chr17
CABP7	chr5	153788597	+	Y	chr5
CARD8	chrX	99922659	+	N	chrX
CASTOR3	chr3	189081695	-	N	chr3
CDH22	chr9	113306486	-	Y	chr9
CFAP69	chr11	10733916	-	N	chr11

Table 1 – continued from previous page

Parental Gene	SIMULATED				FOUN
	Chr	Position	Pol	LINE/SINE	Chr
COL4A3	chr16	46427444	+	N	chr16
COPS2	chr1	38773310	-	Y	chr1
CPNE7	chr9	42228417	+	Y	chr9
DENND2D	chr18	37314709	+	N	chr18
DNAJC27	chr12	60940050	-	N	chr12
EPC2	chr13	94468157	-	N	chr13
EPS8	chr21	26428011	+	N	chr21
ERCC4	chr6	93262920	+	N	chr6
FAAP20	chr9	77384901	-	N	chr9
FAM177B	chr12	130498191	+	N	chr12
FAM71E2	chr2	225319689	+	N	chr2
HAO2	chr14	69901152	+	N	chr14
HEG1	chr3	15517386	-	Y	chr3
HIP1	chr8	75177754	+	Y	chr8
IL1R1	chr8	30386429	-	N	chr8
IQGAP3	chr6	124358143	+	Y	chr6
KIF7	chrX	89251626	-	Y	chrX
LAMP1	chr13	87908197	-	N	chr13
LARS	chr9	64069435	+	Y	chr9
LRRC6	chr4	180728002	-	N	chr4
MACROD2	chr20	18178487	+	N	chr20
MYH10	chr4	186290075	+	Y	chr4
MYH7B	chr13	104241206	+	N	chr13
MYO7A	chr11	14072547	+	N	chr11
NAE1	chr18	74528384	+	Y	chr18
OR14A16	chr1	52758590	+	N	chr1
OR51M1	chr2	37409208	-	N	chr2
OSER1	chr5	53846631	-	Y	chr5
PAFAH1B1	chr15	86208543	+	Y	chr15
PDGFB	chr8	133462380	-	N	chr8
PFKFB2	chr5	36822019	-	N	chr5
PLCB1	chr9	25165703	+	Y	chr9
PNRC1	chr15	48607415	+	N	chr15
PRMT2	chr8	50511539	-	Y	chr8
PRPF18	chr20	51460729	+	Y	chr20
PRSS45P	chr19	5420707	-	Y	chr19
PTPRF	chr19	7227546	+	Y	chr19
RAB18	chr4	10281361	-	N	chr4
RAB5B	chr6	46561322	+	N	chr6
RADX	chr12	117277769	+	N	chr12
RASGEF1C	chr5	115992817	+	N	chr5
RBM4	chr7	101199285	+	Y	chr7
RMDN3	chr3	28655572	-	N	chr3
RNF6	chr4	39797761	-	Y	chr4
SART1	chr2	109317943	+	N	chr2
SDHA	chr4	179658356	+	N	chr4
SEZ6L	chr18	560651	-	Y	chr18
SKP2	chr5	88746051	-	N	chr5

Table 1 – continued from previous page

Parental Gene	SIMULATED				FOUN
	Chr	Position	Pol	LINE/SINE	Chr
SLC9A3	chr4	140369141	-	N	chr4
SMTNL2	chr3	144112843	-	N	chr3
SNRNP27	chrX	13251389	-	N	chrX
STK17B	chrX	36995058	-	Y	chrX
TACO1	chrY	12987416	+	Y	chrY
TMEM63C	chr17	49131966	+	Y	chr17
TMEM95	chr2	234301985	-	Y	chr2
TSFM	chr12	80384739	-	Y	chr12
TUBGCP2	chr1	197233691	+	N	chr1
VIPAS39	chr12	54021508	-	N	chr12
WDR74	chr11	112552782	-	N	chr11
WDR75	chr6	132636317	+	Y	chr6
ZNF136	chr16	59509103	+	Y	chr16
ZNF326	chr8	29273486	-	Y	chr8
ZNF385A	chr12	92752469	-	N	chr12
ZNF431	chr16	88101015	-	N	chr16
ZNF585A	chr18	78888223	-	Y	chr18
ZNF738	chr6	139608184	-	N	chr6
ZNF793	chr9	120420222	+	N	chr9
					RetroCNV events not found by
					Duplica
AC002310.4	chr9	94545202	-	N	chr8:11
AC135178.3	chr7	74794901	-	N	chr7:75
ACSBG2	chr21	43058887	-	N	chr21:6
ADD2	chr3	9759497	+	N	No
AL645922.1	chr6	38626680	-	N	No
C21orf91	chr14	54886570	-	Y	Duplica
CERS1	chr20	41341204	+	N	No
CWC25	chr13	39475646	-	N	No
DHRSX	chr5	166496220	-	Y	Highly
LETM1	chrY	24793930	-	N	8 identi
MALL	chr7	110598366	+	N	No
MRPS7	chr2	1490696	+	N	chr2_K
MTNR1A	chr8	86938090	-	N	chrX, c
NDUFA6	chr10	38060463	+	N	chr10:4
PLAC8	chr9	39225441	+	Y	chr9:61
PTCHD4	chr15	31035142	-	Y	chr15_
SLC44A4	chrY	4417954	+	Y	chrX:9
STON2	chrX	468106	+	N	chrY:4
TAF7	chr22	22384919	-	N	chr22_
TBC1D3F	chr16	65760883	+	Y	No
TRIM40	chr5	45713519	+	N	No

Table 2: sideRETRO capability to identify simulated retroCNVs common (present in all simulated genomes), polymorphic (events present in > 2 genmes) and somatic (events present in only an individual genome).

RetroCNV type	# of simulated events	Found events	%
Common	25	19	76
Polymorphic	50	42	84
Somatic	25	18	72

Table 3: sideRetro performance in identifying simulated retroCNVs. It is shown gene genome coverage, the true positive, false negative, false positive, precision, recall and F1-score considering all simulated retroCNVs (*) and also using those 86 events (**) inserted in mappeable (non ambiguous) genomic regions. These scores are given to the full set of 100 simulated genomes.

Ind	TP	FP	FN*	PPV	TPR (/*)	F1 (/*)
0	38	0	9|5	1.00	0.81|0.88	0.89|0.94
1	36	2	11|7	0.95	0.77|0.84	0.85|0.89
2	33	1	10|6	0.97	0.77|0.85	0.86|0.90
3	35	1	12|5	0.97	0.74|0.88	0.84|0.92
4	29	1	9|5	0.97	0.76|0.85	0.85|0.91
5	37	4	12|5	0.90	0.76|0.88	0.82|0.89
6	45	0	10|6	1.00	0.82|0.88	0.90|0.94
7	37	2	11|5	0.95	0.77|0.88	0.85|0.91
8	32	2	11|5	0.94	0.74|0.86	0.83|0.90
9	33	3	11|5	0.92	0.75|0.87	0.83|0.89
10	34	1	9|5	0.97	0.79|0.87	0.87|0.92
11	37	2	12|5	0.95	0.76|0.88	0.84|0.91
12	30	1	10|5	0.97	0.75|0.86	0.85|0.91
13	43	3	11|5	0.93	0.80|0.90	0.86|0.91
14	38	0	10|6	1.00	0.79|0.86	0.88|0.93
15	31	1	8|5	0.97	0.79|0.86	0.87|0.91
16	30	4	13|6	0.88	0.70|0.83	0.78|0.86
17	39	1	9|5	0.98	0.81|0.89	0.89|0.93
18	37	0	10|5	1.00	0.79|0.88	0.88|0.94
19	39	1	10|6	0.98	0.80|0.87	0.88|0.92
20	39	2	12|6	0.95	0.76|0.87	0.85|0.91
21	42	3	12|5	0.93	0.78|0.89	0.85|0.91
22	39	0	10|6	1.00	0.80|0.87	0.89|0.93
23	41	2	10|5	0.95	0.80|0.89	0.87|0.92
24	43	1	8|5	0.98	0.84|0.90	0.91|0.93
25	41	0	9|6	1.00	0.82|0.87	0.90|0.93
26	43	0	10|6	1.00	0.81|0.88	0.90|0.93
27	34	0	10|5	1.00	0.77|0.87	0.87|0.93
28	38	4	14|7	0.90	0.73|0.84	0.81|0.87
29	36	1	11|6	0.97	0.77|0.86	0.86|0.91
30	47	3	11|5	0.94	0.81|0.90	0.87|0.92
31	43	3	12|5	0.93	0.78|0.90	0.85|0.91
32	38	0	11|5	1.00	0.78|0.88	0.87|0.94
33	34	1	12|6	0.97	0.74|0.85	0.84|0.91
34	35	4	12|6	0.90	0.74|0.85	0.81|0.88

Continued on next page

Table 3 – continued from previous page

Ind	TP	FP	FN*	PPV	TPR (/*)	F1 (/*)
35	43	2	10|6	0.96	0.81|0.88	0.88|0.91
36	41	2	11|6	0.95	0.79|0.87	0.86|0.91
37	38	1	11|6	0.97	0.78|0.86	0.86|0.92
38	34	1	9|5	0.97	0.79|0.87	0.87|0.92
39	39	0	8|5	1.00	0.83|0.89	0.91|0.94
40	35	1	9|5	0.97	0.80|0.88	0.88|0.92
41	33	1	9|5	0.97	0.79|0.87	0.87|0.92
42	39	1	11|7	0.98	0.78|0.85	0.87|0.91
43	37	4	13|7	0.90	0.74|0.84	0.81|0.87
44	39	4	13|6	0.91	0.75|0.87	0.82|0.89
45	35	3	11|6	0.92	0.76|0.85	0.83|0.89
46	31	0	9|5	1.00	0.78|0.86	0.87|0.93
47	36	0	10|5	1.00	0.78|0.88	0.88|0.94
48	40	3	11|6	0.93	0.78|0.87	0.85|0.90
49	34	1	10|5	0.97	0.77|0.87	0.86|0.92
50	41	4	13|6	0.91	0.76|0.87	0.83|0.89
51	34	0	9|5	1.00	0.79|0.87	0.88|0.93
52	36	3	12|5	0.92	0.75|0.88	0.83|0.90
53	39	2	11|5	0.95	0.78|0.89	0.86|0.92
54	47	0	10|6	1.00	0.82|0.89	0.90|0.94
55	36	1	12|5	0.97	0.75|0.88	0.85|0.92
56	40	2	12|6	0.95	0.77|0.87	0.85|0.91
57	41	1	9|5	0.98	0.82|0.89	0.89|0.93
58	40	0	10|5	1.00	0.80|0.89	0.89|0.94
59	34	3	11|6	0.92	0.76|0.85	0.83|0.88
60	35	2	10|5	0.95	0.78|0.88	0.85|0.91
61	38	1	9|5	0.97	0.81|0.88	0.88|0.93
62	30	1	8|5	0.97	0.79|0.86	0.87|0.91
63	38	4	13|6	0.90	0.75|0.86	0.82|0.88
64	43	2	10|5	0.96	0.81|0.90	0.88|0.92
65	46	1	10|6	0.98	0.82|0.88	0.89|0.93
66	41	1	10|6	0.98	0.80|0.87	0.88|0.92
67	37	2	9|5	0.95	0.80|0.88	0.87|0.91
68	44	5	13|6	0.90	0.77|0.88	0.83|0.89
69	36	0	9|5	1.00	0.80|0.88	0.89|0.94
70	42	4	14|7	0.91	0.75|0.86	0.82|0.88
71	44	3	14|7	0.94	0.76|0.86	0.84|0.90
72	41	3	13|6	0.93	0.76|0.87	0.84|0.90
73	34	1	9|5	0.97	0.79|0.87	0.87|0.92
74	42	1	10|5	0.98	0.81|0.89	0.88|0.93
75	37	3	11|5	0.93	0.77|0.88	0.84|0.90
76	34	2	9|5	0.94	0.79|0.87	0.86|0.91
77	37	3	10|5	0.93	0.79|0.88	0.85|0.90
78	38	0	8|5	1.00	0.83|0.88	0.90|0.94
79	40	2	9|5	0.95	0.82|0.89	0.88|0.92
80	35	0	9|5	1.00	0.80|0.88	0.89|0.93
81	40	1	10|6	0.98	0.80|0.87	0.88|0.92
82	41	2	11|7	0.95	0.79|0.85	0.86|0.90
83	39	2	11|6	0.95	0.78|0.87	0.86|0.91

Continued on next page

Table 3 – continued from previous page

Ind	TP	FP	FN*	PPV	TPR (/*)	F1 (/*)
84	40	3	10|6	0.93	0.80|0.87	0.86|0.90
85	36	4	12|5	0.90	0.75|0.88	0.82|0.89
86	37	4	13|6	0.90	0.74|0.86	0.81|0.88
87	32	2	11|5	0.94	0.74|0.86	0.83|0.90
88	42	2	12|7	0.95	0.78|0.86	0.86|0.90
89	34	1	9|5	0.97	0.79|0.87	0.87|0.92
90	41	2	10|5	0.95	0.80|0.89	0.87|0.92
91	45	0	9|6	1.00	0.83|0.88	0.91|0.94
92	39	2	8|5	0.95	0.83|0.89	0.89|0.92
93	39	2	11|6	0.95	0.78|0.87	0.86|0.91
94	34	3	12|5	0.92	0.74|0.87	0.82|0.89
95	44	4	11|5	0.92	0.80|0.90	0.85|0.91
96	36	1	9|5	0.97	0.80|0.88	0.88|0.92
97	39	2	10|5	0.95	0.80|0.89	0.87|0.92
98	48	0	9|6	1.00	0.84|0.89	0.91|0.94
99	40	0	10|6	1.00	0.80|0.87	0.89|0.93
Total	3806	172	1051|551	0.96	0.78|0.87	0.86|0.91

6.2 Real data

The method developed and used by Abyzov et al.[1] relies on exon-exon junction reads to identify **retroCNVs**. In order to increase their candidate's reliability, these authors performed experimental validations (*Abyzov - Table 2*). In summary, the authors. carried out PCR validation for nine putative retroCNVs and for six of them, they found their genomic insertion points (Red blocks). A retroCNV event is, by definition, a retroposition of an mRNA into a genomic region (i.e., it should have an insertion point, otherwise it could

[1] Abyzov, Alexej et al. (2013). Analysis of variable retroduplications in human populations suggests coupling of retrotransposition to cell division. Genome Res, 23:2042-52.

be a distinct retroCNV event, even from the same parental gene). Thus, in order to avoid misleading in data comparison, we selected those retroCNVs events validated by PCR and with a defined genomic insertion point.

YRI trio	CEU trio	Parent gene	Identity to a retro-duplication in the reference	RD support	Insertion point	Identity to retroduplication in HuRef assembly	PCR validation
	✓	*CDC27*	95.6%	✓	—	94.9%	UN
	✓	*LAPTM4B*	93.2%	✓	✓	—	✓
	✓	*TMEM66*	81.7%	✓	✓	—	✓
	✓	*BOD1*	89.0%	—	—	—	✓
	✓	*SKA3*	—	✓	✓	—	✓
	✓	*AL590623.1*	—	—	—	—	NA
✓	✓	*AP3S1*	95.6%	✓	—	99.2%	✓
✓	✓	*CACNA1B*	—	—	✓	—	✓
✓	✓	*TDG*	96.6%	✓	✓	99.8%	✓
✓	✓	*CBX3*	97.5%	✓	✓	99.8%	✓
✓	✓	*MTCH2*	87.5%	—	—	—	✓
✓	✓	*AC131157.1*	96.3%	—	—	—	NA
✓	✓	*BCLAF1*	96.6%	✓	—	99.1%	UN
✓		*TMEM5*	—	✓	✓	—	NA
✓		*ATP9B*	—	✓	—	—	NA
✓		*MFF*	93.5%	✓	✓	—	NA
✓		*AL583842.1*	—	—	—	—	NA

Read depth (RD) tracks supporting predictions are summarized in Supplemental Figures S3–S22. (NA) Not attempted for PCR validation or those for which no suitable primers can be designed. (UN) Cases where expected PCR band was observed, but sequencing of the band failed.

Fig. 2: Highlighted in red: retroCNVs events presenting an insertion point and with PCR validation. Insertion point coordinates were retrieved from Table X, Abyzov et al, Genome Res, 2013.
Highlighted in blue: a lacking of read depth (RD) support to the candidate CACNA1B.

We called retroCNVs using the same 974 individuals from the fourteen (ASW, CEU, CHB, CHS, CLM, FIN, GBR, IBS, JPT, LWK, MXL, PUR, TSI, and YRI) 1000 Genome populations, which are reported in Supplementary _Table S1. Their six retroCNVs with PCR validation and a defined genomic insertion point (presented above, *Abyzov - Table 2*) were used. In summary, our pipeline (sideRETRO) identifies five (83.3%) and misses only one retroCNV (CACNA1B). Regarding the genotyping of retroCNVs shared by Abyzov and us, sideRETRO has a match of 70 genotyping out of 70 (100%), See tables below:

Table 4: RetroCNVs, experimentally validated by PCR and genotyped by Abyzov et al. (2003) and by sideRETRO into individuals from fourteen human populations. TMEM66 (used in Abyzov et al.): now, its official name is SARAF.

Parental Gene	Insertion region (GRCh38; chromosome and position)	
	Abyzov	sideRETRO
CBX3	15:40561954-40561998	15:40561980
LAPTM4B	6:166920412-166920482	6:166920475
TMEM66*	1:191829533-191829591	1:191829594
SKA3	11:108714998-108715054	11:108715020
TDG	12:125316536-125316676	12:125316601
CACNA1B	1:148027670-148027843	

Table 5: Events found by Abzov and sideRETRO are stated as 1/1. Only found by Abyzov: 1/0. Only found by sideRETRO: 0/1. Events absent from Abzov and sideRETRO are stated as 0/0.

Parental Gene	Populations													
	ASW	CEU	CHB	CHS	CLM	FIN	GBR	IBS	JPT	LWK	MXL	PUR	TSI	YRI
CBX3	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1
LAPTM4B	0/0	1/1	0/0	0/0	1/1	1/1	1/1	0/0	0/0	0/0	0/0	1/1	1/1	0/0
TMEM66*	0/0	1/1	0/0	0/0	0/0	1/1	1/1	0/0	0/0	0/0	0/0	1/1	1/1	0/0
SKA3	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1
TDG	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1
CACNA1B	1/0	1/0	1/0	1/0	1/0	1/0	1/0	1/0	1/0	1/0	1/0	1/0	1/0	1/0

Regarding the retroCNV event (parental gene CACNA1B; insertion region: chr1: 147499911-147500084) not identified by sideRETRO:

i) Curiously, Abyzov et al. did not find a good Read Depth Support for it (See above, marked in blue and in their manuscript);

ii) We found that its putative insertion region (GRCh37: chr1:147499911- 147500084; GRCh38: chr1:148,027,670-148,027,843) corresponds to a LTR region (*Part A*- below);

iii) This region has a second (quasi-perfect: only 2 mismatches) hit elsewhere, *Part B*;

iv) Moreover, this second hit is (suspiciously) near to a fixed retrocopy from the same parental gene, CACNA1B (Figure 1C). SideRETRO filters out retroCNVs (i.e., polymorphic) events inserted near a fixed retrocopy from the same parental gene, because they are usually results from false-positive alignments, since their likelihood of being real is very low (roughly = 1 / (genome size x number of genes; haploid genome: 3x109; the number of genes ~ 20k coding genes). Nevertheless, only a further experimental validation may confirm our hypothesis.

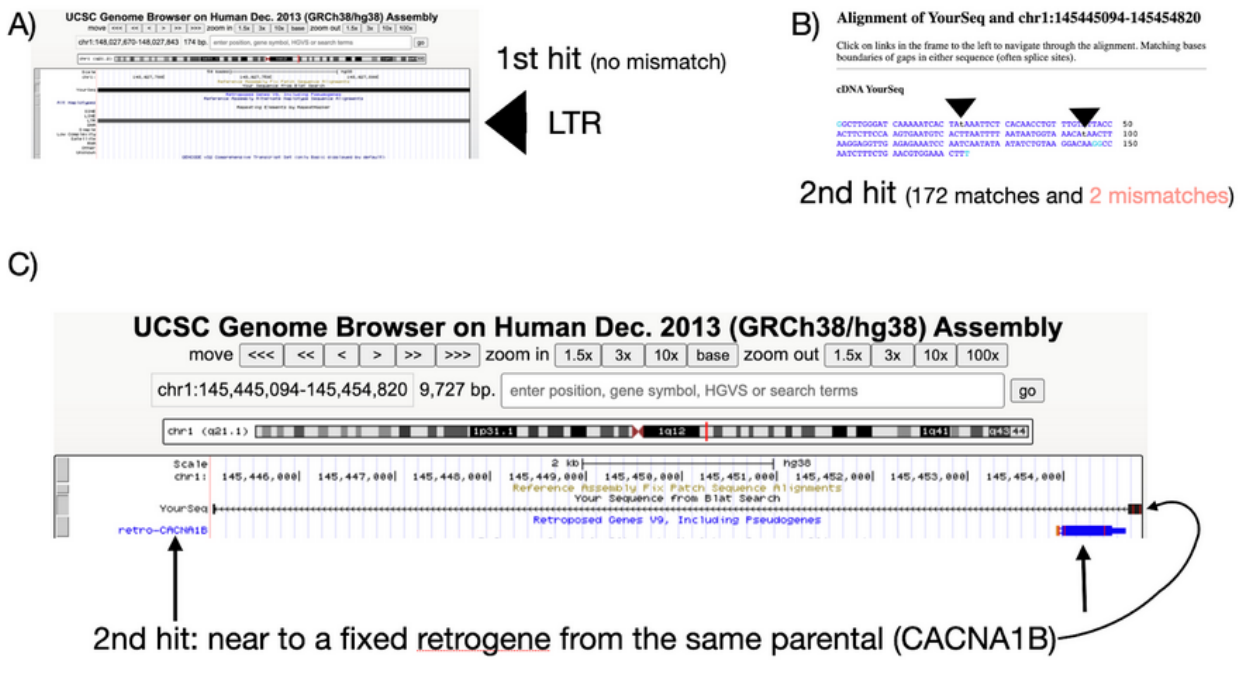


Fig. 3: Genome alignment of the CACNA1B region defined by Abyzov et al. A) genomic alignment of the region defined as the insertion point of CACNA1B (in this case, GRCh38 was used). B) The second hit of this sequence into the genome (only two mismatcher in 174bp). C) The 2nd hit into the genome is near a fixed retrocopy from CACNA1B.

Thus, in summary, regarding the genotyping data, our pipeline presents a very good match ranging from 83.3% (consid-

er-
ing

all events) to 100% (excluding a “suspicious” candidate) against the experimental dataset from an independent group, Abyzov et al. (2013) Gen. Res.

6.3 References and Further Reading

CHAPTER 7

Authors

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

- search