**SVFilter (May 2014)**

**Introduction**

Genomic structural variations (SVs), including large deletions, insertions, inversions, duplications and translocations, constitute an important source of genetic diversity. Recent advances in next-generation sequencing (NGS) technologies and computational algorithms have enabled the genome-wide mapping of SVs at a fine resolution. However, false discovery rate in the current SV discovery programs remains high. Here we have developed the following five filters that can be used to effectively identify false SVs.

* Gap filter
* Norm-ab-ratio-filter
* SNP filter
* Coverage filter (to identify false deletions)
* Read depth filter

**References**

**System requirement and dependencies**

* Linux (required)
* Samtools

### Installation

Installation of SVFilter is straightforward. Just download the SVFilter (wwz server: /home/linyong/SVFilter-1.0.tar.gz 21M) and uncompress the downloaded file.

$ tar -xzvf SVFilter-1.0.tar.gz

It will generate a directory named "SVFilter-1.0" on a Linux machine. The directory includes three subdirectories:

* bin directory: includes all executables.
* input-files directory: includes all necessary input files demonstrated in this manual to test running the SV filter programs
* src directory: includes C++ source codes. Compiling a C++ source code in the following fashion, “g++ -o a.out a.cpp”, should work for all the C++ programs.

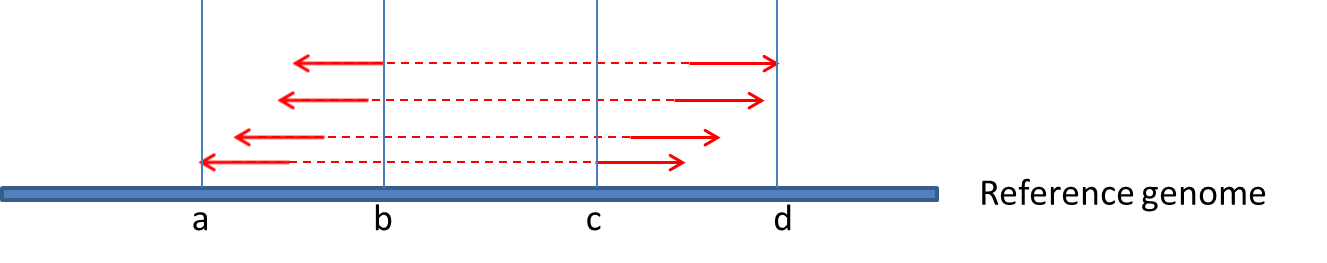
The executables under the “bin” directory were pre-compiled on a 64-bit Linux machine. To run the programs, a convenient way is to add the “bin” directory to the environmental variable PATH.

For a 32-bit Linux machine, user needs to compile the C++ source codes and then moves the executables to the “bin” directory. We included an “install.sh” shell script to run these jobs.

**SV format**

Each of the SV filters requires an SV input file which contains a list of SVs (deletion, insertion, inversion, duplication, etc.). The filter outputs the discarded SVs to the standard output and saves the kept SVs in a new file (see below).

The format of an SV requires 11 fields, as shown in Figure 1. Table 1 shows an example of four SVs. An example of SV input file can also be found under the directory “input-files” with the file name “test.sv.ip”.



|  |  |  |  |
| --- | --- | --- | --- |
| Field # | Field name | Field content | Notes |
| 1 | Left window | Chr1 | Fields 1-4 define an left anchoring window:   * Which chromosome it locates * start and end coordinates (1-based chromosomal position) * orientation of left reads: R for reverse, F for forward |
| 2 | start | a |
| 3 | end | b |
| 4 | Read strand | R |
| 5 | Right window | Chr1 | Fields 5-8 define right anchoring window. |
| 6 | start | c |
| 7 | end | d |
| 8 | Read strand | F |
| 9 | # abnormal pairs | 4 | Number of abnormal read pairs. |
| 10 | Abnormal read IDs | (read\_1\_ID,read\_2\_ID, read\_3\_ID, read\_4\_ID) | In this example, 4 abnormal read pairs comprise left/right anchoring windows. This field lists their read IDs, one for each pair. The order of read IDs does not matter. However, the read IDs must be enclosed within parenthesis and separated by comma. |
| 11 | SV type | LARGE\_DUPLI | A tandem duplication. |

Figure 1. An SV (a tandem duplication) with two anchoring windows, (a, b) and (c, d) on Chr1.

Table 1. SV examples

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Left window | start | end | Read strand | Right window | start | end | Read strand | #  abnormal pairs | Abnormal read names | SV type |
| Chr1 | 884221 | 884482 | F | Chr1 | 886071 | 886334 | R | 6 | (FC42CA5AAXX:5:11:543:682#0,  FC42CA5AAXX:5:64:1039:1736#0,  FC42CA5AAXX:5:101:756:1378#0,  FC42CA5AAXX:5:27:368:1230#0,  FC42CA5AAXX:5:67:1294:1624#0,  FC42CA5AAXX:5:67:1013:1526#0) | DELETION |
| Chr1 | 4848936 | 4849026 | F | Chr1 | 4849100 | 4849192 | R | 3 | (FC42CA5AAXX:5:49:1425:1866#0,  FC42CA5AAXX:5:86:40:593#0,  FC42CA5AAXX:5:61:952:324#0) | INSERTION |
| Chr1 | 7982107 | 7982236 | F | Chr1 | 8306788 | 8306875 | F | 3 | (FC42CA5AAXX:5:93:309:1029#0,  FC42CA5AAXX:5:97:1689:75#0,  FC42CA5AAXX:5:90:254:1603#0) | INVERSION |
| Chr1 | 7029817 | 7029909 | R | Chr1 | 7032304 | 7032391 | F | 4 | (FC42CA5AAXX:5:25:1733:937#0,  FC42CA5AAXX:5:28:736:1157#0,  FC42CA5AAXX:5:52:474:918#0,  FC42CA5AAXX:5:120:442:1038#0) | LARGE\_DUPLI |

**Gap filter**

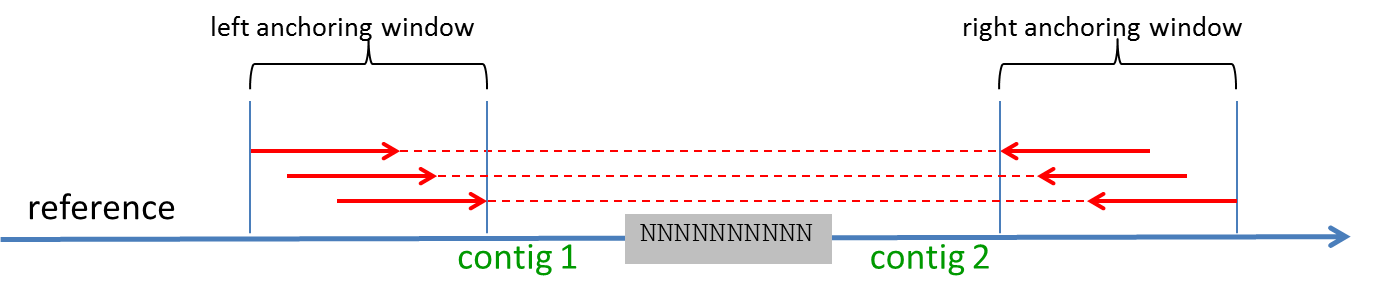


Figure 2. Gap filter.

If an SV spans a genomic region that contains gap(s) (Figure 2), then this SV is discarded. Such SV spans across at least two contigs or even two scaffolds.

Running the program:

$ cd SVFilter-1.0/input-files

$ ../bin/nFilt test.sv.ip genome.fa 1 0.1

The input files “test.sv.ip” and “genome.fa” can be found under the directory “input-files”.

|  |  |  |
| --- | --- | --- |
|  | nFilt | Filter name. |
| Param 1 | test.sv.ip | SV input file. |
| Param 2 | genome.fa | Genome sequence file in fasta format |
| Param 3 | 1 | Cutoff, # of ‘N’ (gap) in the SV region, see below |
| Param 4 | 0.1 | Cutoff, fraction of the SV region being ‘N’, see below |
| Assume the size of an SV is 2534 bp. Within the SV region, 273 bp are ‘N’s (gaps). Because 273 ≥ 1 and 273 / 2534 ≥ 0.1, this SV is considered dubious and discarded. If a user set param\_3 to 1 and param\_4 to a negative value, then an SV will be discarded as long as there is a gap within the SV region. | | |
| Output file | test.sv.ip.DELnnn.filt | Retained SVs after filtering |

**Norm-ab-ratio-filter**

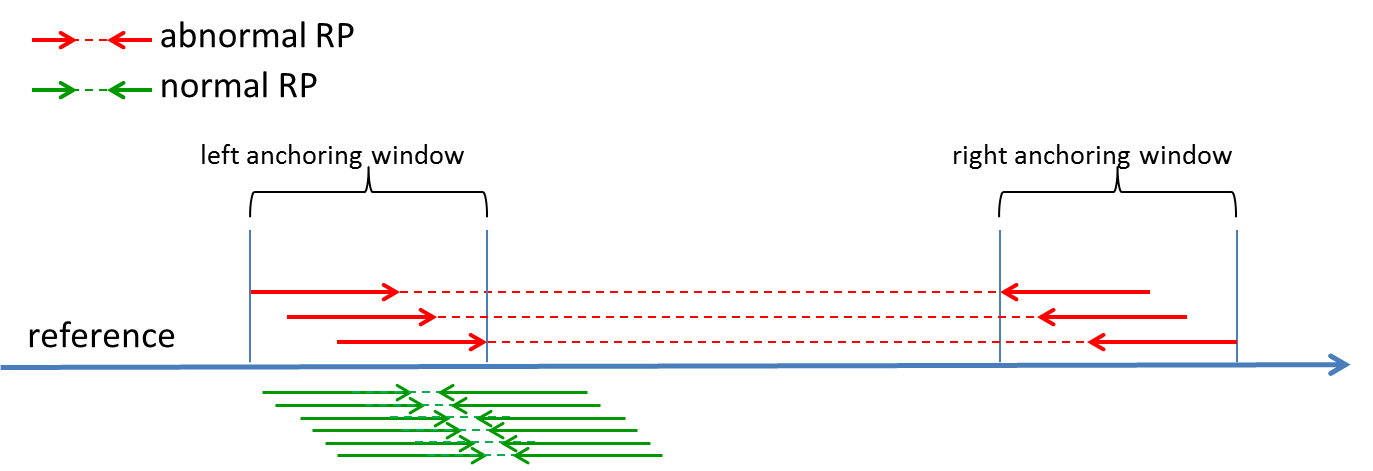


Figure 3. Normal-abnormal-read-ratio filter.

This filter is based on the ratio of normal and abnormal reads (norm-ab-ratio filter). Within an anchoring window where abnormal reads cluster and form an SV, if a significant number of normal reads, which share the same orientation as the abnormal reads, are also present (Figure 3), then this SV is discarded.

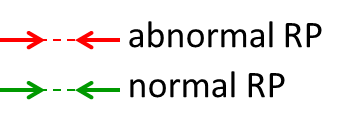
Running the program:

$ cd SVFilter-1.0/input-files

$ ../bin/normRatioFilt test.sv.ip normPair.sam 0.2 75

|  |  |  |
| --- | --- | --- |
|  | normRatioFilt | Filter name. |
| Param 1 | test.sv.ip | SV input file. |
| Param 2 | normPair.sam | Normal paired-end (PE) SAM file. The filter seeks normal reads from this file that are mapped within an anchoring window. |
| Param 3 | 0.2 | Cutoff, ratio between # of normal reads and abnormal reads within an anchoring window. The normal reads must have the same orientations as abnormal reads. See below. |
| Param 4 | 75 | Read length |
| The SV input file specifies # of abnormal read pairs supporting each SV. Within one (or both) of two anchoring windows, if the ratio between # of normal reads and abnormal reads with same orientations exceeds the cutoff value, then the SV is considered as a false positive and discarded. | | |
| Output file | test.sv.ip.normPair.filt | Retained SVs after filtering |

**SNP filter**

****

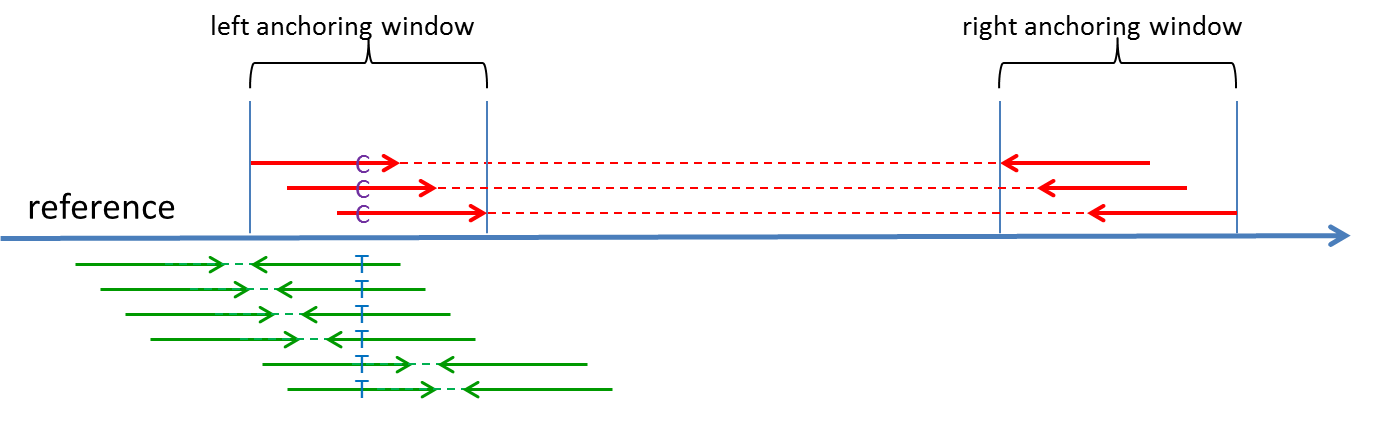
****

Figure 4. SNP filter

This filter is based on SNPs found between the normal and abnormal reads. Within an anchoring window, if SNP(s) or indel(s) can be detected between normal and abnormal reads (Fig 4, C->T substitution between abnormal and normal reads within the left anchoring window), then this SV is removed.

Running the program:

$ cd SVFilter-1.0/input-files

$ export PATH=$PATH:../bin/

$ snpFilt.sh test.sv.ip genome.fa abnorm-pair.sam normPair.sam 2

snpFilt.sh is a shell script. The script calls many samtools utilities, program “script-sam-pileup-ref”, program “fmt.procMulti-getReadName”, program “getSam-ID”, and program “combine2PileFiles”. All these programs are located under the “bin” directory. Thus, in order for the Linux OS to find these executables, user must add the “bin” pathname to the environmental variable PATH. In addition, user must also have samtools installed.

|  |  |  |  |
| --- | --- | --- | --- |
|  | | snpFilt.sh | Filter name |
| Param 1 | | test.sv.ip | SV input file. Abnormal read IDs must be specified in order to use the SNP filter. |
| Param 2 | | genome.fa | Genome sequence file in fasta format |
| Param 3 | | abnorm-pair.sam | Abnormal PE SAM file. The file is used to retrieve abnormal read mapping information based on read IDs. |
| Param 4 | | normPair.sam | Normal PE SAM file. |
| Param 5 | | 2 | Minimal read coverage to detect SNPs between abnormal and normal reads |
| If there is at lease one SNP or indel detected between abnormal and normal reads in one (or both) anchoring window, the SV is considered as a false positive and discarded. | | | |
| Output file | test.sv.ip.snp.filt | | Retained SVs after filtering |

**Coverage filter (only applicable to deletion)**

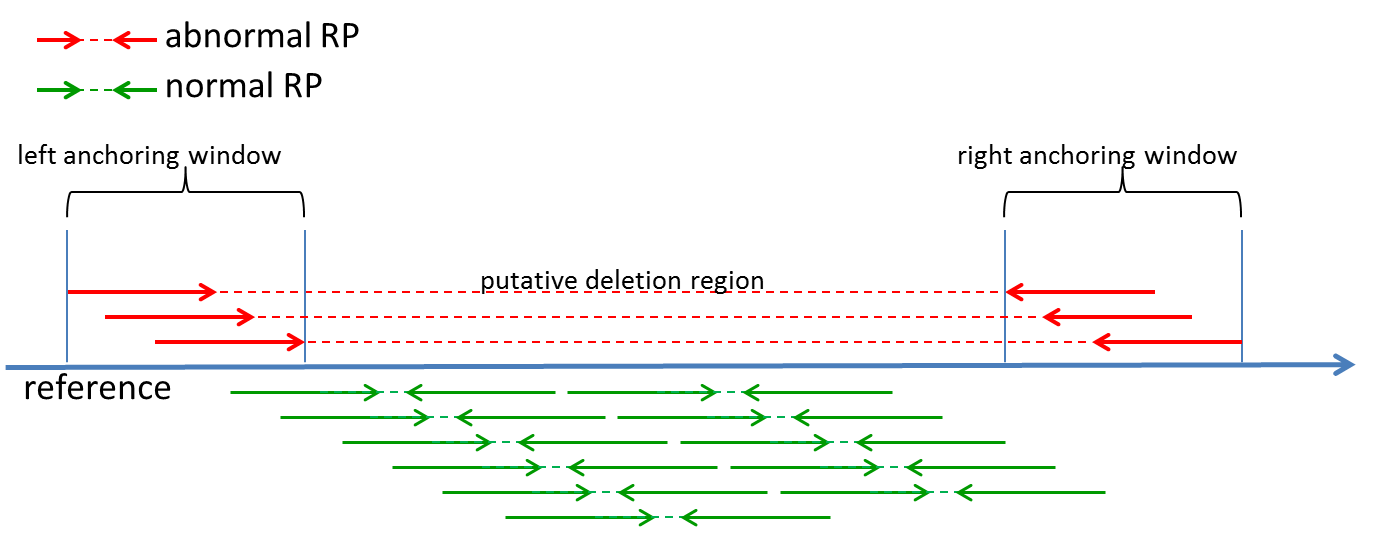


Figure 5. coverage filter.

For a homozygous deletion, there should be no normal reads mapped to the deleted region. If a significant fraction of the deleted region is mapped with normal reads (figure 5), then this candidate deletion is removed.

Running the program:

$ cd SVFilter-1.0/input-files

$ coverFilt test.sv.ip genome.fa test.pileup 6 0.05 2

|  |  |  |
| --- | --- | --- |
|  | coverFilt | Filter name |
| Param 1 | test.sv.ip | SV input file. SV type must be specified in this file. Only SV of the type “DELETION” (all capital letters) will be subject to filtering. |
| Param 2 | genome.fa | Genome sequence file in fasta format |
| Param 3 | test.pileup | Pileup file generated only from normal reads. The file is used to find which parts of the putative deletion region are mapped by normal reads. The pileup file was produced using the “samtools mpileup” utility. |
| Param 4 | 6 | Cutoff, # of base-pair in the deletion region that are mapped by normal reads, see below |
| Param 5 | 0.05 | Cutoff, fraction of the deletion region that are mapped by normal reads, see below |
| Param 6 | 2 | Cutoff, normal read coverage, see below |
| Assume the deletion length is 2534 bp. Within the deletion region, 273 bp are also covered by normal reads with at least 2x coverage. Because 273 ≥ 6 and 273 / 2534 ≥ 0.05, this deletion is considered as a false positive and discarded. | | |
| Output file | test.sv.ip.DELnormPileup.filt | Retained SVs after filtering |

**Read depth filter (comparing read depth in a duplication region with the genome-wide read depth)**

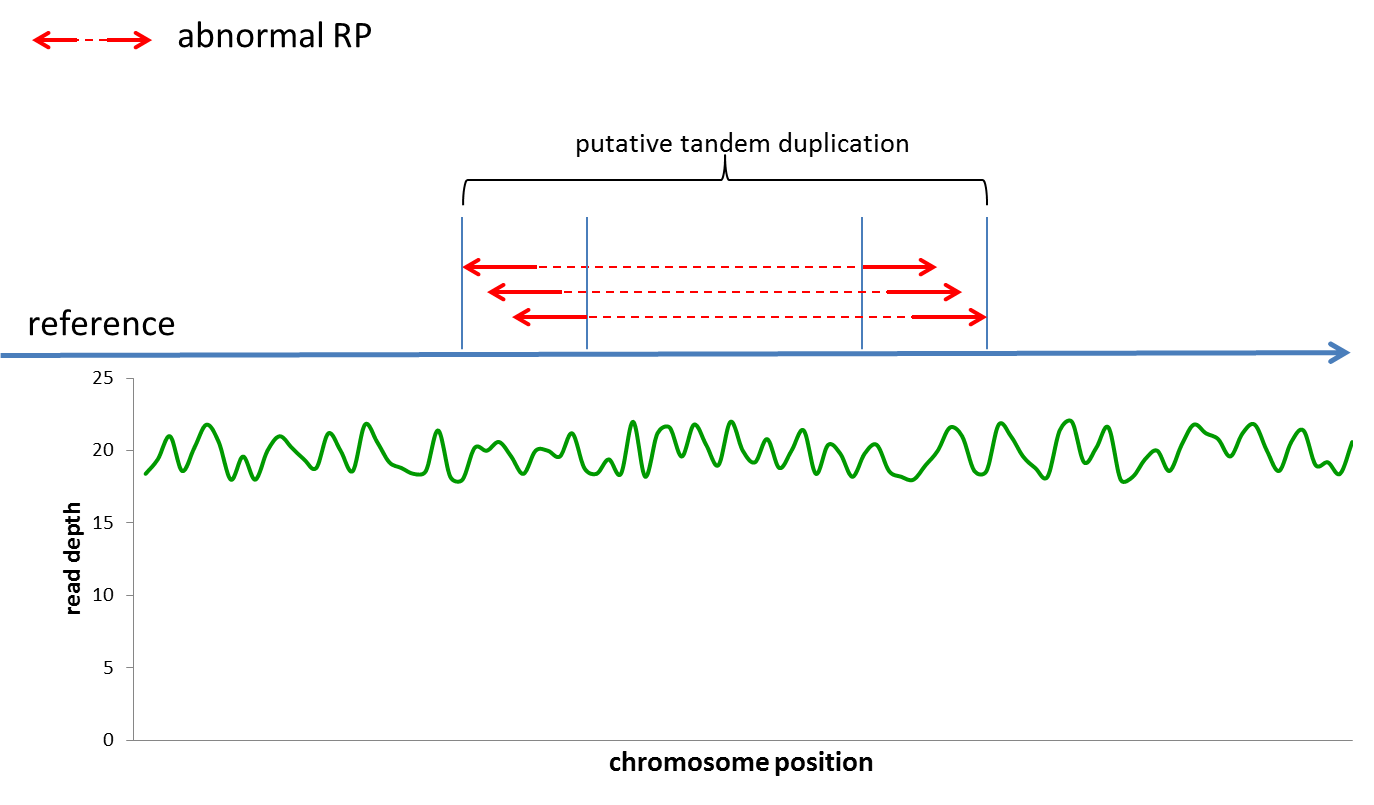
****

Figure 6. read depth filter.

For a tandem duplication, read depth in the duplicated segments should be significantly higher than the genomic background. If the average read depth over the duplicated segments is not significantly higher than the genome-wise average (figure 6), then the predicted tandem duplication is rejected.

Running the program:

$ cd SVFilter-1.0/input-files

$ seqDepthFilt test.sv.ip chr-length test.pileup 1.5 1.5

|  |  |  |
| --- | --- | --- |
|  | seqDepthFilt | Filter name |
| Param 1 | test.sv.ip | SV input file. SV type must be specified in this file. Only SV of the type “LARGE\_DUPLI” (all capital letters) will be subject to filtering. |
| Param 2 | chr-length | The file lists length for each chromosome / contig. Each line in the file, delimited by tab, consists of chromosome ID and its length. An example of the file, “chr-length”, is included under the “input-files” directory. |
| Param 3 | test.pileup | Pileup file generated only from normal reads. The file is used to calculate read depths. |
| Param 4 | 1.5 | Cutoff, ratio between average of read depths in the duplication region and the average over entire genome, see below |
| Param 5 | 1.5 | Cutoff, ratio between median of read depths in the duplication region and the genome-wide median, see below |
| The program calculates read depth average and median in a duplication region. It also calculates genome-wide read depth average and median. If the ratio between the duplication-region average and genome-wide average > cutoff, and the ratio between the duplication-region median and genome-wide median > cutoff, then this duplication is considered as a true positive and retained. Otherwise, the duplication is discarded. | | |
| Output file | test.sv.ip.seqDepth.filt | Retained SVs after filtering |

**Q & A**

1. I obtained a set of SVs by using breakpoint-based approach. Can I use your filters?

In this case, start/end positions of left anchoring window will be collapsed to left breakpoint. start/end positions of right anchoring window will be set to right breakpoint (as shown in the table below). Because both anchoring windows are collapsed into a single line, norm-ab-ratio filter and SNP filter can not be used. For the same reason, it is not necessary to specify read IDs. You can just put arbitrary non-empty word in the field (such as “NA”, “readID”, “anyWord”). However, gap filter, coverage filter and read depth filter are still applicable for your SVs.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Chr1 | 884482 | 884482 | F | Chr1 | 886071 | 886071 | R | 6 | anyWord | DELETION |

2. Do I have to provide read IDs in the SV input file?

Only SNP filter requires read IDs because it has to derive genotype information from the abnormal reads. You can still use the other filters without specifying read IDs. In this case, put arbitrary non-empty word in the field (such as “NA”, “readID”, “anyWord”).

3. I ran the following command, nFilt /A/B/my-sv.ip genome.fa 1 0.1, I can’t find the output file which stores the retained SVs.

The output file, my-sv.ip.DELnnn.filt, is located under the /A/B/ directory.