VarBen Manual

Introduction

VarBen is a software to add SNV/Indel, CNV and SV to BAM files, used for testing mutation callers and pipelines.

Software Dependencies

1. samtools (http://samtools.sourceforge.net/)
2. pysam (http://code.google.com/p/pysam/ or pip install pysam)
3. bwa/tmap/novoalign

Known bugs and limitation

VarBen is under rapid development driven by suggesting and bug reports from the mutation calling community.

1. Currently, we are working on testing a new version of mutation editor for Ion Torrent plantform.
2. There is a bug in parallel function, the user cannot stop software via ctrl-C during it is running in parallel mode.

Function 1. Mutation editor (muteditor.py)

```
usage: muteditor.py [-h] -m MUTFILE -b BAMFILE -r REFFASTA -o OUTDIR
                    --alignerIndex ALIGNERINDEX [-p PROCESS] [--seqer SEQER]
                    [-g] [--aligner ALIGNER] [--haplosize HAPLOSIZE]
                    [--mindepth MINDEPTH] [--minmutreads MINMUTREADS]
                    [--snpfrac SNPFRAC] [--minmapq MINMAPQ] [--multmapfilter]
                    [--diffcover DIFFCOVER] [--floworder FLOWORDER]
                    [--libkey LIBKEY] [--barcode BARCODE] [--tag]
```

Edit bamfile to spike in SNV, Indel, Complex, Substitution

Function 2. SV/CNV editor (sveditor.py)

```
usage: sveditor.py [-h] -m SVFILE -b BAMFILE -r REFFASTA -l READLENGTH -o
                   OUTDIR --alignerIndex ALIGNERINDEX [-p PROCESS]
                   [--seqer SEQER] [-g] [--aligner ALIGNER]
                   [--mindepth MINDEPTH] [--minmutreads MINMUTREADS]
                   [--minmapq MINMAPQ] [--multmapfilter]
                   [--floworder FLOWORDER] [--libkey LIBKEY]
                   [--barcode BARCODE] [--tag]
```

Edit bam file to spike in SV

Local optional arguments (only used in muteditor.py)

`-m MUTFILE, --mutfile MUTFILE`

Target regions to try and spike in a point mutation.

There are four types of snv/indel included in the software: `snv`, `ins`(insertion), `del`(deletion), `Sub`(Complex mutation). The file format is shown as below.

```
#Chrom Start End AlleleFrequency Type AlternativeSequence

chr1 899778 899778 0.9 snv T

chr1 3712508 3712508 0.9 snv T

chr1 1158637 1158638 0.9 ins TAG

chr1 3397038 3397039 0.9 ins AGGTAG

chr1 6533124 6533126 0.9 del .

chr1 7910946 7910956 0.9 del .

chr7 55242467 55242481 0.3 Sub TTC ### Complex indel format: EGFR, c.2237_2251>TTC(p.E746_T751>VP)
```

How to determine the start and end position?

- For single nucleotide variant, the start and end site should be the same position in the genome.
- For short sequence insertion, the start and end site should has 1 base difference.
- For short sequence deletion, the start and end site shows the sequence start and end position which will exclude from the sequencing reads.
- For Complex mutation, we want to using a short **sequence A** to instead of a **sequence B**, we should put the **sequence B**'s start and end position in our mutation file.

Local optional arguments (only used in sveditor.py)

`-m SVFILE, --svfile SVFILE`

Target regions to try and spike in a SV or CNV.

There are six types of SV included in the software: `inv`(inversion), `del`(deletion), `dup`(duplication), `trans_chrom` (whole arm translocate chromosome), `trans_balance`(balanced translocation chromosome), `trans_unbalance` (insertional translocation chromosome).

del & inv format

```
#chrom  start end type  AF

chrX  12994966  12996009  del 0.6

chrX  20172336  20176010  del 0.6
```

```
chrX  105121310 105134706   del 0.6
chrX  108614726 108616334   del 0.6
chrX  13703890  14134046  inv 0.6
chrX  19975999  20064786  inv 0.6
chrX  32391049  32794255  inv 0.6
chrX  40994338  41012689  inv 0.6
```

dup format

```
#chrom  start end type  AF  dup_num
chr1  15808448  15814030  dup 0.6 3
chr1  16076907  16086182  dup 0.6 4
chr1  23665443  23711586  dup 0.6 3
chr1  28057278  28081157  dup 0.6 3
```

trans_chrom & trans_balance & trans_unbalance format

```
#CHR1 CHR1_start  CHR1_end  type  AF  CHR2  CHR2_start  CHR2_end
chr10 7059511 7059511 trans_chrom 0.5 chr19 17396810  17396810
chr19 17327977  17327977  trans_chrom 0.5 chr3  186528041 186528041
chr3  107598967 107598967 trans_chrom 0.5 chr7  38371959  38371959
chr1  31561816  31561816  trans_chrom 0.5 chr6  41297838  41297838
chr2  29754284  29754947  trans_balance 0.5 chr2  42522695  42523089
chr10 43608984  43609308  trans_unbalance 0.5 chr6  117640981 117640982
```

There are two types of CNV included in the software: `gain`and `loss`. The file format is shown as below.

```
#chrom  start end type  AF  cnv_type
chrX 66764255 66950650 cnv 2.5 gain
chr20 52186265 52200826 cnv 2 loss
```

`-l READLENGTH, --readlength READLENGTH`

The read length of BAM file.

Global optional arguments

`-h, --help`

To show the help message and exit.

`-b BAMFILE, --bamfile BAMFILE`

A BAM file to spike in mutations, the bam file should be sorted and indexed, the user also needs to provide a BAM index `.bai` file with the same prefix name as BAM file. By default, the software considers the BAM file consists by entirely paired-end reads, if a user needs to spike mutation in a BAM file which consists by single-end reads, they need using the `-single` option.

`-r REFFASTA, --reffasta REFFASTA`

Genome reference, FASTA file with corresponding index `.fai` file which is generated by Samtools. The target BAM file should be generated by the same reference file used in this option, especially the chromosome names and lengths in the reference FASTA must be the same as in the BAM header. This FASTA file is used to create pseudo reads near the editing poisition.

`-o OUTDIR, --outdir OUTDIR`

A output directory name for edited bam file and other information.

`--alignerIndex ALIGNERINDEX`

The index database sequences in the FASTA format of aligner. For example if the aligner is bwa, then bwa index should be provided. This FASTA file is called by the external aligner.

`-p PROCESS, --process PROCESS`

Parallel mode: process number (default = 1)

`--seqer SEQER`

Define the seqer: illumina, life, BGI (default is `illumina`)

`-g, --single`

To declare that the input bam is single-ended (default is `False`)

`--aligner ALIGNER`

Choose an aligner from bwa, novoalign and tmap (default is `bwa`).

`--haplosize HAPLOSIZE`

The size of haplotype block to consider when adding more than 1 proximal mutation. (default = 0)

For example, if two SNVs are spiked in 5bp apart and `-haplosize` is 5 or greater, the two SNVs will be on the same haplotype (i.e. share the same reads for reads covering both positions)

`--mindepth MINDEPTH`

The minimum depth of reads position which could be used as a spike in site. (default = 30)

For instance, if one spike in position reads depth is 25X (there are only 25 reads covered this position), the VarBen software will drop this spike in position automatically due to the reads depth is not enough to add in any mutation.

`--minmutreads MINMUTREADS`

The minimum number of reads to be edited in one position (default = 5).

VarBen will calculate the number of mutated reads by the allele frequency and the total number of reads in the position. If the mutation reads number is less than 5, the software will not add in any mutation in this position.

`--snpfrac SNPFRAC`

To avoid spike any mutatoin on top of exisiting heterozygous alleles, the heterozygous allele fraction set to 0.2 (default = 1)

`--minmapq MINMAPQ`

A read mapping quality less than `MINMAPQ` will not be considered to edit (default 20).

`--multmapfilter`

Any multi-mapped reads will not be considered to edit (default is `True`).

`--diffcover DIFFCOVER`

The coverage difference allowed between the input BAM and output BAM (default 0.9).

`--floworder FLOWORDER`

If seqer is `life`, a flower order of life sequence should be provided.

`--libkey LIBKEY`

If seqer is `life`, a libkey of life sequence should be provided.

`--barcode BARCODE`

If seqer is `life`, a barcode of life sequence should be provided.

`--tag`

Add tag to edited reads (default False).