BAMSurgeon: Methods for spike-in mutations on BAM files

Adam D. Ewing (adam.ewing@mater.uq.edu.au)

August 17, 2018

1 Introduction

1.1 Software Dependencies

BAMSurgeon requires the following software packages be available:

1. samtools/wgsim/tabix (`http://samtools.sourceforge.net/`)
2. pysam (`http://code.google.com/p/pysam/` or `pip install pysam`)
3. bwa (`http://bio-bwa.sourceforge.net/`)
4. velvet (`http://www.ebi.ac.uk/~zerbino/velvet/`)
5. exonerate (`https://github.com/nathanweeks/exonerate`)
6. picard (`http://http://picard.sourceforge.net/`)

For install instructions please see the included README.md or have a look at the Dockerfile.

1.2 Known bugs and limitations

BAMSurgeon is under rapid development driven by suggesting and bug reports from the mutation calling community. The following is an incomplete TODO list of things needing attention:

1. Adding SVs above 30kb is very slow
2. Simulated SVs cannot be specified exactly
3. Heterozygous mutations in SV regions tend to increase or decrease in allele fraction due to one or the other haplotype being assembled and used as the basis for read coverage simulation. This can lead to false-positive mutations in SV regions, so we suggest masking these regions or simulating SVs and SNVs/INDELs on seperate BAMs.

2 Adding SNVs to existing BAM alignments (addsnv)

2.1 Usage

```
usage: addsnv.py [-h] -v VARFILENAME -f BAMFILENAME -r REFFASTA -o OUTBAMFILE
                 [-s SNVFRAC] [-m MUTFRAC] [-n NUMSNVS] [-c CNVFILE]
                 [-d COVERDIFF] [-z HAPLOSIZE] [-p PROCS]
```

```
                [--picardjar PICARDJAR] [--mindepth MINDEPTH]
                [--maxdepth MAXDEPTH] [--minmutreads MINMUTREADS]
                [--avoidreads AVOIDREADS] [--nomut] [--ignoresnps]
                [--ignoreref] [--force] [--insane] [--single]
                [--maxopen MAXOPEN] [--requirepaired] [--tagreads]
                [--skipmerge] [--ignorepileup] [--aligner ALIGNER]
                [--alignopts ALIGNOPTS] [--tmpdir TMPDIR] [--seed SEED]

adds SNVs to reads, outputs modified reads as .bam along with mates

optional arguments:
  -h, --help            show this help message and exit
  -v VARFILENAME, --varfile VARFILENAME
                        Target regions to try and add a SNV, as BED
  -f BAMFILENAME, --bamfile BAMFILENAME
                        sam/bam file from which to obtain reads
  -r REFFASTA, --reference REFFASTA
                        reference genome, fasta indexed with bwa index _and_
                        samtools faidx
  -o OUTBAMFILE, --outbam OUTBAMFILE
                        .bam file name for output
  -s SNVFRAC, --snvfrac SNVFRAC
                        maximum allowable linked SNP MAF (for avoiding
                        haplotypes) (default = 1)
  -m MUTFRAC, --mutfrac MUTFRAC
                        allelic fraction at which to make SNVs (default = 0.5)
  -n NUMSNVS, --numsnvs NUMSNVS
                        maximum number of mutations to try (default: entire
                        input)
  -c CNVFILE, --cnvfile CNVFILE
                        tabix-indexed list of genome-wide absolute copy number
                        values (e.g. 2 alleles = no change)
  -d COVERDIFF, --coverdiff COVERDIFF
                        allow difference in input and output coverage
                        (default=0.9)
  -z HAPLOSIZE, --haplosize HAPLOSIZE
                        haplotype size (default = 0)
  -p PROCS, --procs PROCS
                        split into multiple processes (default=1)
  --picardjar PICARDJAR
                        path to picard.jar, required for most aligners
  --mindepth MINDEPTH   minimum read depth to make mutation (default = 10)
  --maxdepth MAXDEPTH   maximum read depth to make mutation (default = 2000)
  --minmutreads MINMUTREADS
                        minimum number of mutated reads to output per site
  --avoidreads AVOIDREADS
                        file of read names to avoid (mutations will be skipped
                        if overlap)
```

```
--nomut               dry run
--ignoresnps          make mutations even if there are non-reference alleles
                      sharing the relevant reads
--ignoreref           make mutations even if the mutation is back to the
                      reference allele
--force               force mutation to happen regardless of nearby SNP or
                      low coverage
--insane              ignore sanity check enforcing input read count =
                      output read count in realignment
--single              input BAM is simgle-ended (default is paired-end)
--maxopen MAXOPEN     maximum number of open files during merge (default
                      1000)
--requirepaired       skip mutations if unpaired reads are present
--tagreads            add BS tag to altered reads
--skipmerge           final output is tmp file to be merged
--ignorepileup        do not check pileup depth in mutation regions
--aligner ALIGNER     supported aligners:
                      backtrack,mem,novoalign,gsnap,STAR,bowtie2,tmap,bwakit
--alignopts ALIGNOPTS
                      aligner-specific options as comma delimited list of
                      option1:value1,option2:value2,...
--tmpdir TMPDIR       temporary directory (default=addsnv.tmp)
--seed SEED           seed random number generation
```

2.2 Description

Single nucleotide changes are introduced to an existing BAM alignment using `addsnv.py` as outlined in Figure 1. Input consists of a list of locations where SNVs will be made, a target BAM alignment, and a reference genome indexed using bwa (preferably the same genome used to align the reads in the BAM file).

2.3 Input

Mutation input (-v/–varfile) Input to addsnv.py (-v/–varfile) is a three column format plus an extra optional column for the desired variant allele fraction (VAF). Input coordinates are **1-based**, meaning they will match between input to -v/–varfile, VCF, and samtools mpileup. Note that `addindel.py` uses a 1-based coordinate scheme. The default VAF is specified via -m/–mutfrac. For example:

```
22 34166720 34166720 0.25
22 33908770 33908770 0.15
22 33714964 33714964 0.5
22 33769483 33769483 0.75
22 33958087 33958087 0.01
22 34264774 34264774 0.25
22 33702084 33702084 0.2
```

Specific bases can also be specified using a fifth input column:

```
22 34166720 34166720 0.25 A
22 33908770 33908770 0.15 T
22 33714964 33714964 0.5  G
22 33769483 33769483 0.75 C
```

The above input will cause addsnv to attempt to add SNVs at the specified locations, using the VAF specified in the fourth column if available, otherwise defaulting to 0.5 or whatever is specified by -m/–mutfrac (e.g. the last three lines). If a position is specified that spans more than one base, the position of the SNV input will be chosen at random from that interval.

BAM Alignment input (-f/–bamfile) BAMs ideally should adhere to SAM specification i.e. they should validate via picardś ValidateSamFile. BAMs should be sorted in coordinate order and indexed. BAMs should consist either of entirely paired-end reads or entirely single-end (fragment) reads using the –single option, mixing paired and single reads yields unpredictable results. Currently, bwa backtrack (`bwa aln`) and bwa mem are supported, although it should be straightforward enough to add further aligners through modification of the `remap_paired` function. In practice we have been able to spike mutations into RNA-seq MapSplice BAMs, for instance.

Reference genome (-r/–reference) The reference genome should be the **same as that used to create the target BAM file**, specifically the chromosome names and lengths in the reference FASTA must be the same as in the BAM header. The reference must be indexed for bwa (`bwa index`) and indexed with samtools (`samtools faidx`).

Avoiding existing SNPs with -s/–snpfrac (optional, default=1.0) In an attempt to preserve haplotype structure and avoid confusion by spiking mutations on top of existing heterozygous alleles, the -s/–snpfrac cutoff may be useful. For a selected mutation site S and specified cutoff f starting from the leftmost base L of the leftmost read containing S and ending at the rightmost base R of the rightmost read containing S, the minor allele frequency M for every intervening base is calculated for every base in the interval $[L, R]$. If $M < f$ for all bases in $[L, R]$ the mutation at S is allowed, and skipped otherwise. Note, do not confuse this with -m/–mutfrac, which is the target variant allele frequency for mutations unless overridden in the mutation input file. By default this function is turned off (threshold minor allele frequency set to 1.0). Setting this too low can yield few successful mutations as sequence errors will inhibit successful spike-in, we recommend a setting of 0.1 for Illumina GA/HiSeq reads.

CNV file (-c/–cnvfile) (optional) It is possible to simulate adding mutations to a single haplotype for copy-number variant regions. For example, if a region has absolute copy number 3 (instead of 2), a mutation into one haplotype should have an apparent VAF of roughly 0.33 instead of 0.5. This optional file should be in BED-3 format plus a fourth column representing the absolute copy number for the region defined by the chromosome/contig, start, and end fields. The genome is assumed to be diploid thus an absolute copy number of 2 will not adjust the variant allele fractions. Absolute copy numbers below 2 will adjust VAFs higher and absolute copy numbers above 2 will adjust VAFs correspondingly lower.

Additional options

- -d/–coverdiff : threshold on output coverage divided by input coverage for mutation sites. Mutations can cause alignments to change, if this is undesired set this high, if this is part of the fun, set this low.

- -z/–haplosize : specifies size of haplotype block to consider when adding more than 1 proximal mutation. For instance, if two SNVs are spiked in 10bp apart and –haplosize is 10 or greater, the two SNVs will be on the same haplotype (i.e. share the same reads for reads covering both positions)
- -p/–procs : BAMSurgeon programs currently spawn a new process for each spikein, to speed things along the processes can be run concurrently with the number of simultaneous jobs set by this parameter.
- –picardjar : path to picard.jar provided by picard tools, this is required for the some aligner options.
- –mindepth : minimum depth at sites for mutations, sites with fewer reads in pileup column will be dropped. Default is 10 reads.
- –maxdepth : sites with greater than this number of covering reads will be dropped. Default is 2000 reads. (May need to increase for targeted sequencing data)
- –minmutreads : minimum number of mutated reads per site. If this level of coverage cannot be obtained, mutations will be dropped.
- –ignoresnps: ignores existing non-reference bases at the mutation site, required if the goal is to change a non-reference allele back to reference
- –ignoreref: required if the goal is to change a non-reference allele back to reference
- –avoidreads : addsnv and addindel can be passed a list of read names to avoid when making mutations. Any sites selected that overlap these reads will be dropped. This feature exists as a means to ensure mutations across multiple runs do not overlap.
- –force : force the addition of a mutation even if it fails thresholds set by –mindepth and -c/–coverdiff. Not recommended for most use cases.
- –insane: ignores sanity checking number of input reads versus number of output reads. This seems to be required for some aligners or situations where unaligned reads are dropped or secondary alignments are added.
- –single : covered above, assume reads are single-end (i.e. fragment reads).
- –maxopen : set the maximum number of open files (check your system limit with `ulimit -a`)
- –requirepaired : drop mutations where a paired end in the region cannot be located. However, it is probably better to correct the problem in the BAM file than to resort to using this option.
- –skipmerge : instead of replacing the mutated and realigned reads back into the original BAM, which is normally the final step, leave the addsnv.
- –aligner : choose one of the supported aligners, default is backtrack (for bwa backtrack). Additional aligners can be added through modification of bs/aligners.py, and I try to accomodate requests for widely-used aligners if they are not already implemented. Aligner specific options are handled through –alignopts

- –alignopts : aligner-specific options are specified in a comma-delimited list as option1:value1,option2:value2,et Required options are checked by the the function `checkoptions` in `bs/aligners.py`. For example, if –aligner novoalign is specified, –alignopts novoref:¡path to novoalign reference¿ should also be specified.
- –tagreads : adds a 'BS' tag to the reads altered by bamsurgeon, useful in debugging
- –seed : sets a random seed for reproducability
- options not listed here are either new and the manual has not been updated as yet, or they are there for debugging and will be removed in the future.

3 Adding INDELs to existing BAM alignments (addindel)

3.1 Usage

```
usage: addindel.py [-h] -v VARFILENAME -f BAMFILENAME -r REFFASTA -o
                   OUTBAMFILE [-s SNVFRAC] [-m MUTFRAC] [-n NUMSNVS]
                   [-c CNVFILE] [-d COVERDIFF] [-p PROCS]
                   [--picardjar PICARDJAR] [--mindepth MINDEPTH]
                   [--maxdepth MAXDEPTH] [--minmutreads MINMUTREADS]
                   [--avoidreads AVOIDREADS] [--nomut] [--det] [--force]
                   [--insane] [--single] [--maxopen MAXOPEN] [--requirepaired]
                   [--aligner ALIGNER] [--alignopts ALIGNOPTS] [--tagreads]
                   [--skipmerge] [--ignorepileup] [--tmpdir TMPDIR]
                   [--seed SEED]

adds INDELs to reads, outputs modified reads as .bam along with mates

optional arguments:
  -h, --help            show this help message and exit
  -v VARFILENAME, --varfile VARFILENAME
                        Target regions to try and add a SNV, as BED
  -f BAMFILENAME, --bamfile BAMFILENAME
                        sam/bam file from which to obtain reads
  -r REFFASTA, --reference REFFASTA
                        reference genome, fasta indexed with bwa index _and_
                        samtools faidx
  -o OUTBAMFILE, --outbam OUTBAMFILE
                        .bam file name for output
  -s SNVFRAC, --snvfrac SNVFRAC
                        maximum allowable linked SNP MAF (for avoiding
                        haplotypes) (default = 1)
  -m MUTFRAC, --mutfrac MUTFRAC
                        allelic fraction at which to make SNVs (default = 0.5)
  -n NUMSNVS, --numsnvs NUMSNVS
                        maximum number of mutations to try (default: entire
                        input)
  -c CNVFILE, --cnvfile CNVFILE
```

```
                      tabix-indexed list of genome-wide absolute copy number
                      values (e.g. 2 alleles = no change)
  -d COVERDIFF, --coverdiff COVERDIFF
                      allow difference in input and output coverage
                      (default=0.1)
  -p PROCS, --procs PROCS
                      split into multiple processes (default=1)
  --picardjar PICARDJAR
                      path to picard.jar
  --mindepth MINDEPTH   minimum read depth to make mutation (default = 10)
  --maxdepth MAXDEPTH   maximum read depth to make mutation (default = 2000)
  --minmutreads MINMUTREADS
                      minimum number of mutated reads to output per site
  --avoidreads AVOIDREADS
                      file of read names to avoid (mutations will be skipped
                      if overlap)
  --nomut             dry run
  --det               deterministic base changes: make transitions only
  --force             force mutation to happen regardless of nearby SNP or
                      low coverage
  --insane            ignore sanity check enforcing input read count =
                      output read count in realignment
  --single            input BAM is simgle-ended (default is paired-end)
  --maxopen MAXOPEN     maximum number of open files during merge (default
                      1000)
  --requirepaired     skip mutations if unpaired reads are present
  --aligner ALIGNER     supported aligners:
                      backtrack,mem,novoalign,gsnap,STAR,bowtie2,tmap,bwakit
  --alignopts ALIGNOPTS
                      aligner-specific options as comma delimited list of
                      option1:value1,option2:value2,...
  --tagreads          add BS tag to altered reads
  --skipmerge         final output is tmp file to be merged
  --ignorepileup      do not check pileup depth in mutation regions
  --tmpdir TMPDIR       temporary directory (default=addindel.tmp)
  --seed SEED           seed random number generation
```

3.2 Description

Short insertions and deletions (commonly referred to as INDELs) can be added to BAM files using addindel.py. The options are essentially the same as for addsnv.py, but the syntax of the input file is slightly different and, importantly is **0-based** (i.e. constent with BED files):

```
22 33694303 33694304 0.50 INS ATGGC
22 33815273 33815274 0.25 INS CG
22 33920490 33920491 0.33 INS ATTTGCTTAGCTGAGGGCTTAGGCTTAGCATGCGTA
22 33944542 33944543 0.50 DEL
22 34006643 34006653 0.50 DEL
```

```
22 33958087 33958090 0.40 DEL
```

The first four columns are the same as for addsnv.py input but the fourth column (VAF) is manditory. The fifth column must be either 'INS' or 'DEL' (for insertion or deletion). For insertions there must be a sixth column specifying the sequence to be inserted at the position indicated in the second column (the position in the third column does not matter for insertions). In the above example, the first three columns direct addindel.ppy to (1) add an insertion of "ATGGC" at chromosome 22 position 33694303 with 50% VAF, (2) add an insertion of "CG" at position 338152273 with 25% VAF, and (3) add an insertion of "ATTTGC..." with 33% VAF.

For deletions, the sequence from the start to the end (columns two and three) is deleted. Lines 4,5, and 6 above indicate deletions of 1 bp, 10bp, and 3 bp, respectively.

4 Adding SVs to existing BAM alignments (addsv.py)

4.1 Usage

```
usage: addsv.py [-h] -v VARFILENAME -f BAMFILENAME -r REFFASTA -o OUTBAMFILE
                [-l MAXLIBSIZE] [-k KMERSIZE] [-s SVFRAC]
                [--minctglen MINCTGLEN] [-n MAXMUTS] [-c CNVFILE]
                [--ismean ISMEAN] [--issd ISSD] [-p PROCS] [--inslib INSLIB]
                [--delay DELAY] [--noref] [--aligner ALIGNER]
                [--alignopts ALIGNOPTS] [--tagreads] [--skipmerge]
                [--keepsecondary] [--debug] [--tmpdir TMPDIR] [--seed SEED]
                [--allowN]

adds SVs to reads, outputs modified reads as .bam along with mates

optional arguments:
  -h, --help            show this help message and exit
  -v VARFILENAME, --varfile VARFILENAME
                        whitespace-delimited target regions for SV spike-in,
                        see manual for syntax
  -f BAMFILENAME, --bamfile BAMFILENAME
                        sam/bam file from which to obtain reads
  -r REFFASTA, --reference REFFASTA
                        reference genome, fasta indexed with bwa index _and_
                        samtools faidx
  -o OUTBAMFILE, --outbam OUTBAMFILE
                        .bam file name for output
  -l MAXLIBSIZE, --maxlibsize MAXLIBSIZE
                        maximum fragment length of seq. library
  -k KMERSIZE, --kmer KMERSIZE
                        kmer size for assembly (default = 31)
  -s SVFRAC, --svfrac SVFRAC
                        allele fraction of variant (default = 1.0)
  --minctglen MINCTGLEN
                        pad input intervals out to a minimum length for contig
                        generation (default=3000)
```

```
-n MAXMUTS            maximum number of mutations to make
-c CNVFILE, --cnvfile CNVFILE
                      tabix-indexed list of genome-wide absolute copy number
                      values (e.g. 2 alleles = no change)
--ismean ISMEAN       mean insert size (default = estimate from region)
--issd ISSD           insert size standard deviation (default = estimate
                      from region)
-p PROCS, --procs PROCS
                      split into multiple processes (default=1)
--inslib INSLIB       FASTA file containing library of possible insertions,
                      use INS RND instead of INS filename to pick one
--delay DELAY         time delay between jobs (try to avoid thrashing disks)
--noref               do not perform reference based assembly
--aligner ALIGNER     supported aligners: backtrack,mem,novoalign
--alignopts ALIGNOPTS
                      aligner-specific options as comma delimited list of
                      option1:value1,option2:value2,...
--tagreads            add BS tag to altered reads
--skipmerge           do not merge spike-in reads back into original BAM
--keepsecondary       keep secondary reads in final BAM
--debug               output read tracking info to debug file, retain all
                      intermediates
--tmpdir TMPDIR       temporary directory (default=addsv.tmp)
--seed SEED           seed random number generation
--allowN              allow N in contigs, replace with A and warn user
                      (default: drop mutation)
```

4.2 Description

Larger structural variants (insertions, deletions, duplications, inversions, and compound rearrangements) are added to existing BAM alignments using `addsv.py` as described in Figure 3. Input consists of a list of regions where SVs will be made along with a specification of each variant, and as with addsnv and addindel, a target BAM alignment, and a reference genome indexed using bwa are required.

4.3 Input

The input mutation list consists of four columns: chromosome, start of region, end of region, and a controlled-vocabulary description of the mutation. A mutation will not be made if the largest contig obtained from local assembly of the specified region is less than 3 times the maximum expected library size (specified by `-l/--maxlibsize`). The mutation description starts with either `INS, DEL, DUP, TRN`, or `INV` for insertion, deletion, duplication, transduction, and inversion, respectively and is followed by mutation-specific options. An example follows.

```
22 34171055 34184700 INS ACTAACCTGCACAATGTGCACATGTACCCTAAAACTTAGAGTATAATAAAAAAAA 13 AA^TTTT 0.8
22 33607508 33607508 TRN 22 34314981 34314981 0.9
22 33871043 33884754 DEL 0.9 0.75
22 34071043 34084754 INV 0.9
```

```
22 34271043 34284754 DUP 3 0.9
```

For translocations (`TRN`), the above example will join the 6kbp region centered on 22:33607508 and the 6kpb region centered on 22:34314981. The variant allele fraction (VAF) is specified in the last column (0.9 in the above example).

For insertions, `INS` should be followed by either a FASTA file containing the sequence to be inserted, by the nucleotide sequence itself, or by RND if a library of potential insertion sequences is passed in FASTA format via the –inslib option. For example, `INS ATG` would insert the sequence ``ATG'' in the middle of the largest contig obtained from the specified region, while `INS LINE1.fa` would insert the sequence in the FASTA-formatted file `LINE1.fa` into the largest contig. Many insertions include target site duplications (TSDs), so to simulate this an integer value can be added in the fifth column of insertion entries specifying the TSD length. Endonuclease motifs are another common feature of insertion breakpoints, so to simulate this, a preferred cut site can be defined in the 6th column of an insertion entry with the syntax NNNˆNNN where 'NNN' denotes sequence of any length composed of characters (A, T, G, C) with the actual cut site in the motif specified by the caret (ˆ). An optional seventh column can be used to specifiy the VAF, 0.8 in the example above.

For deletions, `DEL` should be followed by the fraction of the largest contig to be deleted, where 1.0 indicates a deletion of the largest assembled region minus one library length on each end. e.g. DEL 0.5. This is optionally followed by specifying the per-mutation VAF (0.75 in the example above). Inversions (`INV`) have no further options aside from an optional VAF column - the region inside the largest contig will be inverted. Duplications (`DUP`) have two optional parameters, an integer specifying the number of times the sequence of the largest contig should be duplicated, e.g. `DUP 2` specifies the region is duplicated twice, and a VAF column. In general, we recommend that short insertions and deletions (those shorter than one read length) be created with addindel.py instead of addsv.py as addindel.py does not require reconstruction of the region through sequence assembly and is thus less error-prone.

Compound variants are also possible by chaining a number of mutations together in a semicolon-delimited list, e.g. `DUP 1;DEL 0.5;INS AAATCC;INV` would duplicate the region inside the largest contig once, delete half the width of the region, insert the sequence `AAATCC`, and invert the region (not currently supported with transductions). Good luck!

4.4 Controlling SV VAF

Variant allele fraction for `addsv.py` can be controlled either via the `-c/--cnvfile` option (see Section 2.3) or through addition of a column in the input file (float value between 0.0 and 1.0). Note that all other optional columns must be specified for the VAF column to be effective, which means the input format for `addsv.py` really needs an overhaul.

4.5 Output

Where mutations add sequence (e.g. duplications), new reads are created that will be added to the original BAM, and where mutations remove sequence (e.g. deletions) those reads are maked as excluded (excluded read names are recorded in the file specified by `-x/--excluded`). Excluded reads are not carried over from the original BAM to the mutated BAM, creating the copy number effect associated with deletion.

Output consists of a BAM file containing the spike-in mutations and a directory of log files describing which bases were changed in which reads (one log per mutation) and the local assemblies for each region where a mutation was made. Currently, the directory containing output logs for

SV spike-ins can be transformed into VCF format using the script `makevcf_sv.py` in the `etc/` subdirectory. The spike-in process may be done concurrently through the use of the `-p/--procs` option. Here are some extended descriptions for additional options not mentioned yet:

- -s/–svfrac : equivalent to -m/–mutfrac for addindel.py and addsnv.py
- -k/–kmer : kmer size to be fed to velvet, which is used for assembly
- -n : only attempt this many mutations
- –ismean/–issd : the mean and standard deviation of the library size should be estimated from the original bam and specified through these arguments. This can be accomplished a number of ways, I suggest CollectInsertSizeMetrics from the picard tools.
- –inslib : a FASTA file can be specified and insertions can be tagged with RND instead of a sequence or file. If this is done, a sequence will be chosen at random from the FASTA file specified by –inslib and inserted.
- –skipmerge : same meaning as for addsnv/addindel
- –allowN : useful if desired mutations are being dropped due to one or more N bases present in the assembled contig

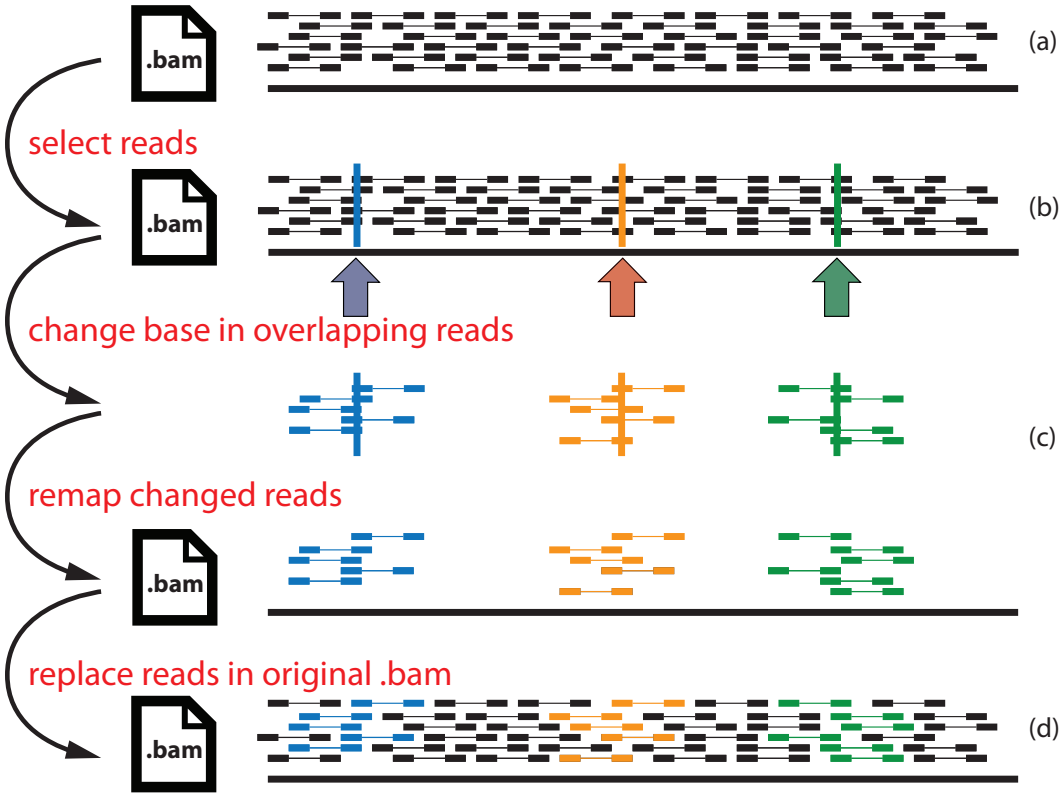

Figure 1: Method for adding SNVs to existing BAM alignments. Starting with the original BAM alignment (a) and a list of positions (b), reads overlapping the chosen positions are selected the target base change is made (c). Altered reads are re-mapped the the reference genome using bwa, and the modified, realigned reads replace the unmodified versions in the original BAM (d).

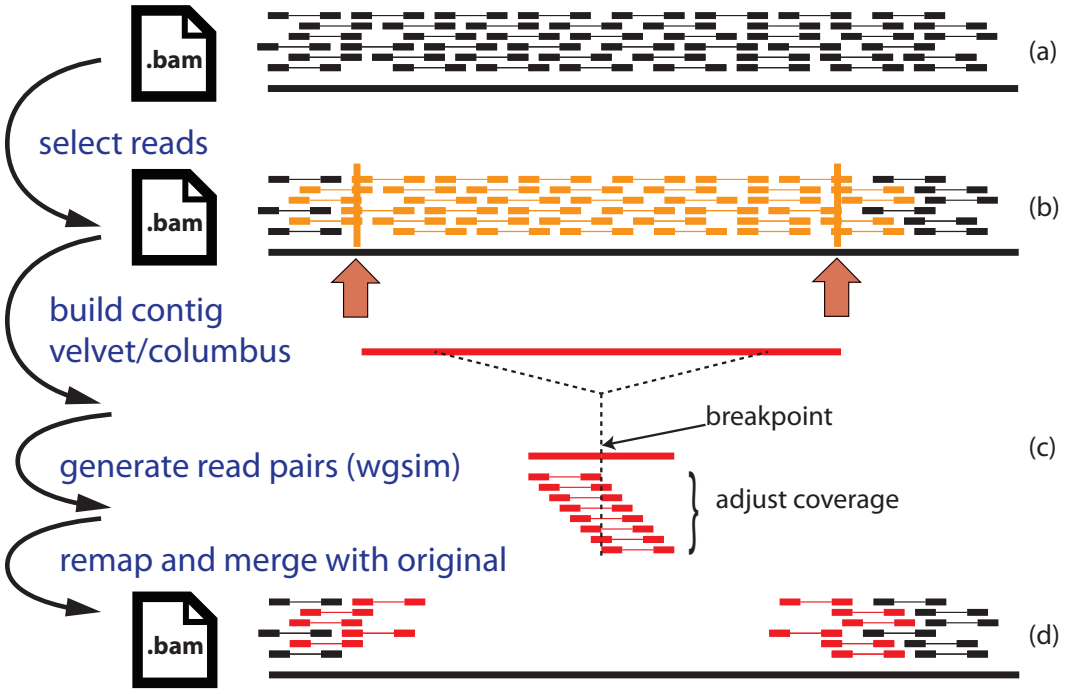

Figure 2: Method for adding multi-nucleotide variants (indels and SVs) to existing BAMs: deletion example. Starting with the original BAM alignment (a) regions, e.g. (b), are assembled using Velvet/Columbus (`http://www.ebi.ac.uk/ zerbino/velvet/`) and the desired mutation(s) are created in the largest contig. Read coverage is simulated over the contig using wgsim (`https://github.com/lh3/wgsim`, also included with samtools) with parameters `-e 0 -r 0 -R 0` to suppress additional mutations, other parameters are set based on the input BAM and desired coverage. Simulated coverage is scaled based on whether the mutation is adds sequence (duplication) or removes sequence (deletions), and replaced into the original BAM while excluded reads are removed from the original BAM (d).

5 Testing

The `test/` directory contains scripts useful for testing bamsurgeon's various modes, the primary ones are as follows:

5.1 Testing addsnv.py

The script test/test_snv.sh may be run with no parameters to display a usage statement:

```
usage: ./test_snv.sh <number of SNPs> <number of threads> <reference indexed with bwa index>
```

Or for testing with BWA MEM:

```
./test_snv_bwamem.sh <number of SNPs> <number of threads> <reference indexed with bwa index> <p
```

The number of SNPs can be up to 100 and are selected from the file `test_data/random_snvs.txt`, number of threads must be specified, use 1 to run on a single core, and the final argument must be a path to a human genome (GRCh37) reference indexed with `bwa index -a bwtsw`. The reference **should not** use the 'chr' prefix on the chromosome names.

5.2 Testing addindel.py

The script test/test_sv.sh may be run with no parameters to display a usage statement:

```
usage: ./test_indel.sh <number of threads> <reference indexed with bwa index>
```

5.3 Testing addsv.py

The script test/test_sv.sh may be run with no parameters to display a usage statement:

```
usage: ./test_sv.sh <number of threads> <reference indexed with bwa index>
```

6 Other Considerations

6.1 System Requirements

For each process (specified by -p/–procs) add*.py will require 4GB RAM. So for 8 concurrent mutation processes, you should have 32GB RAM. System requirements may decrease in the future as we find ways to improve efficiency. For more recent versions of bwa, it is possible to load the reference index into shared memory with 'bwa shm' which seems to improve performance.

6.2 Alignments

Reads should be re-aligned using the same aligner and parameters as were used to create the original BAM file. By default, bamsurgeon assumes alignments were performed using bwa aln. Specific parameters that are not currently exposed on the command line may be altered through modification of the functions `remap_paired` (for paired reads) or `remap_single`. Supported aligners currently include: bwa aln, bwa mem, novoalign, gsnap, STAR, bowtie2, tmap, bwakit.

7 Acknowledgements

I would like to thank and acknowledge everyone who has submitted a pull request or bug report through github, the DREAM Somatic Mutation Calling challenge, or by other means. This software was initially written when I was in the Haussler Lab at UCSC, and continues to be maintained in my current position at the Mater Research Insitute - University of Queensland. The Boutros Lab (OICR) has been and continues to be an important source of support, testing, and interesting use cases for Bamsurgeon.