TEBreak: Generalised insertion detection

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

July 19, 2015

1 Introduction

1.1 Software Dependencies and Installation

TEBreak requires the following software packages be available:

1. samtools (`http://samtools.sourceforge.net/`)
2. bwa (`http://bio-bwa.sourceforge.net/`)
3. LAST (`http://last.cbrc.jp/`)
4. minia (`http://minia.genouest.org/`)

Please run the included setup.py to check that external dependencies are installed properly and to install the required python libraries:

```
python setup.py build
python setup.py install
```

2 Testing / example run

2.1 Prepare reference genome

Any number of variations on the human reference genome are available. To obtain the version used to generate the included example BAM please do the following:

```
wget ftp://ftp.ncbi.nlm.nih.gov/sra/reports/Assembly/GRCh37-HG19_Broad_variant/Homo_sapiens_as
bwa index Homo_sapiens_assembly19.fasta
samtools faidx Homo_sapiens_assembly19.fasta
```

2.2 Run tebreak.py

Starting in the TEBreak root directory:

```
tebreak/tebreak.py \
-b test/data/example.ins.bam \
-r Homo_sapiens_assembly19.fasta \
-i test/data/example.region.bed \
--pickle test/example.pickle \
--detail_out test/example.tebreak.detail.out
```

Output should be similar to the following:

```
2015-07-19 17:16:05,185 commandline: tebreak/tebreak.py -b test/data/example.ins.bam -r Homo_sa
2015-07-19 17:16:05,186 loading bwa index /home/adam/dev/genomes/broad/Homo_sapiens_assembly19
2015-07-19 17:16:05,192 loaded.
2015-07-19 17:16:05,244 chunk count: 1
2015-07-19 17:16:05,248 Processing chunk: 4:57459002-57496002 ...
2015-07-19 17:16:05,248 Chunk 4:57459002-57496002: Parsing split reads from bam(s): test/data/
2015-07-19 17:16:05,364 Chunk 4:57459002-57496002: Building clusters from 151 split reads ...
2015-07-19 17:16:05,369 Chunk 4:57459002-57496002: Building breakends...
2015-07-19 17:16:05,616 Chunk 4:57459002-57496002: Mapping 74 breakends ...
2015-07-19 17:16:06,312 Chunk 4:57459002-57496002: Building insertions...
2015-07-19 17:16:06,333 Chunk 4:57459002-57496002: Processing and filtering 40 potential inser
2015-07-19 17:16:06,456 Chunk 4:57459002-57496002: Postprocessing 6 filtered insertions, trying
2015-07-19 17:16:08,623 Chunk 4:57459002-57496002: Summarising insertions ...
2015-07-19 17:16:08,623 Finished chunk: 4:57459002-57496002
2015-07-19 17:16:08,635 Pickled to test/example.pickle
```

2.3 Run resolve.py

The second step is to resolve putative insertions, in this case human transposable element insertions, using resolve.py:

```
tebreak/resolve.py \
-p test/example.pickle \
-i lib/teref.human.fa \
--detail_out test/example.resolve.detail.out \
-v > test/example.tab.txt
```

You should see something like the following:

```
2015-07-19 17:25:39,039 resolve.py called with args: tebreak/resolve.py -p test/example.pickle
2015-07-19 17:25:39,039 loading pickle: test/example.pickle
2015-07-19 17:25:39,044 finished loading test/example.pickle
2015-07-19 17:25:39,044 raw candidate count: 6
2015-07-19 17:25:39,044 prefiltered candidate count: 6
2015-07-19 17:25:39,044 Create LAST db for tebreak_refs/teref.human.fa ...
2015-07-19 17:25:39,093 submitted 0 candidates, last uuid: f7b95d73-07e9-4b6d-868b-6553c5a7c5f
2015-07-19 17:25:39,120 Samtools indexing /tmp/tebreak.ref.ALU:AluYa5.f7b95d73-07e9-4b6d-868b-
2015-07-19 17:25:39,128 Create BWA db for /tmp/tebreak.ref.ALU:AluYa5.f7b95d73-07e9-4b6d-868b-
2015-07-19 17:25:39,253 Samtools indexing /tmp/tebreak.ref.L1:L1preTa.11f42f35-c098-4d56-b9eb-
2015-07-19 17:25:39,259 Create BWA db for /tmp/tebreak.ref.L1:L1preTa.11f42f35-c098-4d56-b9eb-
2015-07-19 17:25:39,370 Samtools indexing /tmp/tebreak.ref.L1:L1Ta.d9fa11ac-dd69-4267-8774-dc8
2015-07-19 17:25:39,376 Create BWA db for /tmp/tebreak.ref.L1:L1Ta.d9fa11ac-dd69-4267-8774-dc8
2015-07-19 17:25:39,502 Samtools indexing /tmp/tebreak.ref.ALU:AluYa5.4c1fa9a7-5fed-44c3-8ca3-
2015-07-19 17:25:39,508 Create BWA db for /tmp/tebreak.ref.ALU:AluYa5.4c1fa9a7-5fed-44c3-8ca3-
2015-07-19 17:25:39,643 Samtools indexing /tmp/tebreak.ref.ALU:AluYa5.f805e9b6-6922-47cf-bef3-
2015-07-19 17:25:39,649 Create BWA db for /tmp/tebreak.ref.ALU:AluYa5.f805e9b6-6922-47cf-bef3-
2015-07-19 17:25:39,745 Samtools indexing /tmp/tebreak.ref.SVA:SVA_F.1d70cd96-ed12-40ed-ba2c-3
2015-07-19 17:25:39,751 Create BWA db for /tmp/tebreak.ref.SVA:SVA_F.1d70cd96-ed12-40ed-ba2c-3
```

2.4 Filter the output table

Please see the section on filtering for details.

First you'll need to build the hg19 mappability index:

```
cd lib
./human_mappability.sh
cd ..
```

Once that is completed, filter the results from resolve.py:

```
scripts/filter_hg19.py \
/home/adam/repos/tebreak \
test/example.tab.txt > test/example.filtered.tab.txt
```

If all went well, test/example.filtered.tab.txt should contain evidence for 5 transposable element insertions (2 L1s and 3 Alus). If it does not, please double check that all prerequisites are installed, read the rest of this document, try again, and e-mail me at adam.ewing@mater.uq.edu.au with error messages if you are still not having any luck.

3 Insertion site discovery (tebreak.py)

3.1 Usage

```
usage: tebreak.py [-h] -b BAM -r BWAREF [-p PROCESSES] [-c CHUNKS]
                  [-i INTERVAL_BED] [-D MAXD] [--min_minclip MIN_MINCLIP]
                  [--min_maxclip MIN_MAXCLIP]
                  [--min_sr_per_break MIN_SR_PER_BREAK]
                  [--min_consensus_score MIN_CONSENSUS_SCORE] [-m MASK]
                  [--rpkm_bam RPKM_BAM] [--max_fold_rpkm MAX_FOLD_RPKM]
                  [--max_ins_reads MAX_INS_READS]
                  [--min_split_reads MIN_SPLIT_READS]
                  [--min_prox_mapq MIN_PROX_MAPQ]
                  [--max_N_consensus MAX_N_CONSENSUS]
                  [--exclude_bam EXCLUDE_BAM]
                  [--exclude_readgroup EXCLUDE_READGROUP]
                  [--max_bam_count MAX_BAM_COUNT]
                  [--insertion_library INSERTION_LIBRARY]
                  [--map_tabix MAP_TABIX] [--min_mappability MIN_MAPPABILITY]
                  [--tmpdir TMPDIR] [--pickle PICKLE]
                  [--detail_out DETAIL_OUT] [--wg_rpkm] [--no_rpkm]
                  [--no_shared_mem]

Find inserted sequences vs. reference

optional arguments:
  -h, --help            show this help message and exit
  -b BAM, --bam BAM     target BAM(s): can be comma-delimited list
  -r BWAREF, --bwaref BWAREF
```

```
                        bwa/samtools indexed reference genome
  -p PROCESSES, --processes PROCESSES
                        split work across multiple processes
  -c CHUNKS, --chunks CHUNKS
                        split genome into chunks (default = # processes),
                        helps control memory usage
  -i INTERVAL_BED, --interval_bed INTERVAL_BED
                        BED file with intervals to scan
  -D MAXD, --maxD MAXD  maximum value of KS D statistic for split qualities
                        (default = 0.8)
  --min_minclip MIN_MINCLIP
                        min. shortest clipped bases per cluster (default = 3)
  --min_maxclip MIN_MAXCLIP
                        min. longest clipped bases per cluster (default = 10)
  --min_sr_per_break MIN_SR_PER_BREAK
                        minimum split reads per breakend (default = 1)
  --min_consensus_score MIN_CONSENSUS_SCORE
                        quality of consensus alignment (default = 0.95)
  -m MASK, --mask MASK  BED file of masked regions
  --rpkm_bam RPKM_BAM   use alternate BAM(s) for RPKM calculation: use
                        original BAMs if using reduced BAM(s) for -b/--bam
  --max_fold_rpkm MAX_FOLD_RPKM
                        ignore insertions supported by rpkm*max_fold_rpkm
                        reads (default = 10)
  --max_ins_reads MAX_INS_READS
                        maximum number of reads per insertion call (default =
                        1000)
  --min_split_reads MIN_SPLIT_READS
                        minimum total split reads per insertion call (default
                        = 4)
  --min_prox_mapq MIN_PROX_MAPQ
                        minimum map quality for proximal subread (default =
                        10)
  --max_N_consensus MAX_N_CONSENSUS
                        exclude breakend seqs with > this number of N bases
                        (default = 4)
  --exclude_bam EXCLUDE_BAM
                        may be comma delimited
  --exclude_readgroup EXCLUDE_READGROUP
                        may be comma delimited
  --max_bam_count MAX_BAM_COUNT
                        maximum number of bams supporting per insertion
  --insertion_library INSERTION_LIBRARY
                        for pre-selecting insertion types
  --map_tabix MAP_TABIX
                        tabix-indexed BED of mappability scores
  --min_mappability MIN_MAPPABILITY
                        minimum mappability (default = 0.5; only matters with
                        --map_tabix)
  --tmpdir TMPDIR       temporary directory (default = /tmp)
  --pickle PICKLE       pickle output name
  --detail_out DETAIL_OUT
                        file to write detailed output
  --wg_rpkm             force calculate rpkm over whole genome
  --no_rpkm             do not filter sites by rpkm
  --no_shared_mem
```

3.2 Description

Insertion sites are discovered through clustering and scaffolding of clipped reads. Additional support is obtained through local assembly of discordant read pairs, if applicable. Input requirements are minimal, consisting of one of more indexed BAM files and the reference genome corresponding to the alignments in the BAM files(s). Many additional options are available and recommended to improve performance and/or sensitivity.

3.3 Input

BAM Alignment input (-b/–bam) 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 may consist of either paired-end reads, fragment (single end) reads, or both. Multiple BAM files can be input in a comma-delimited list.

Reference genome (-r/–bwaref) 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`).

Pickled output (–pickle) Output data in python's pickle format, meant for input to other scripts including resolve.py and picklescreen.py (in /scripts). Default is the basename of the input BAM with a .pickle extension.

Detailed human-readable output (–detail_out) This is a file containing detailed information about consensus reads, aligned segments, and statistics for each putative insertion site detected. Note that this is done with minimal filtering, so these should not be used blindly. Default filename is tebreak.out

Additional options

- -p/–processes : Split work across multiple processes. Parallelism is accomplished through python's multiprocessing module. If specific regions are input via -i/–interval_bed, these intervals will be distributed one per process. If a whole genome is to be analysed (no -i/–interval_bed), the genome is split into chunks, one per process, unless a specific number of chunks is specified via -c/–chunks.

- -i/–interval_bed : BED file specifying intervals to be scanned for insertion evidence, the first three columns must be chromosome, start, end.

- -D/–maxD : Maximum value of D statistic from Kolmogorov-Smirnov test used to check similarity between the distribution of mapped base qualities versus clipped base qualities for a soft-clipped read (default = 0.8).
- –min_minclip : the shortest amount of soft clipping that will be considered (default = 3, minimum = 2).
- –max_minclip : For a given cluster of clipped reads, the greatest number of bases clipped form any read in the cluster must be at least this amount (default = 10).
- –min_sr_per_break : minimum number of split (clipped) reads required to form a cluster (default = 1)
- –min_consensus_score : minimum quality score for the scaffold created from clipped reads (default = 0.95)
- -m/–mask : BED file of regions to mask. Reads will not be considered if they fall into regions in this file.
- –rpkm_bam : use alternate BAM file(s) for RPKM calculation for avoiding over-aligned regions (useful for subsetted BAMs).
- –max_fold_rpkm : reject cluster if RPKM for clustered region is greater than the mean RPKM by this factor (default = 10)
- –max_ins_reads : maximum number of reads per insertion call (default = 1000)
- –min_split_reads : minimum total split (clipped) read count per insertion (default = 4)
- –min_prox_mapq : minimum mapping quality for proximal (within-cluster) alignments (default = 10)
- –max_N_consensus : exclude reads and consensus breakends with greater than this number of N (undefined) bases (default = 4)
- –exclude_bam : only consider clusters that do not include reads from these BAM(s) (may be comma-delimited list)
- –exclude_readgroup : only consider clusters that to not include reads from these readgroup(s) (may be comma-delimited list)
- –max_bam_count : set maximum number of BAMs involved per insertion
- –insertion_library : pre-select insertions containing sequence from specified FASTA file (not generally recommended but may improve running time in some instances)
- –map_tabix : tabix-indexed BED of mappability scores. Generate for human with script in lib/human_mappability.sh.
- –min_mappability : minimum mappability for cluster (default = 0.5; only effective if –map_tabix is also specified)
- –tmpdir : directory for temporary files (default = /tmp)
- –wg_rpkm : calculate average RPKM using entire genome instead of regions in -i/–interval_bed

- –no_rpkm : disable filtering pileup depth by RPKM.
- –no_shared_mem : By default, tebreak.py will try to use `bwa shm` to load the reference genome into shared memory as this is much more efficient for multiprocessing. This option disables shared memory.

4 Resolution of specific insertion types (resolve.py)

4.1 Usage

```
usage: resolve.py [-h] -p PICKLE [-t PROCESSES] -i INSLIB_FASTA
                  [-m FILTER_BED] [-v] [--max_bam_count MAX_BAM_COUNT]
                  [--min_ins_match MIN_INS_MATCH] [--minmatch MINMATCH]
                  [--annotation_tabix ANNOTATION_TABIX]
                  [--map_tabix MAP_TABIX] [--refoutdir REFOUTDIR]
                  [--skip_align] [--use_rg] [--keep_all_tmp_bams]
                  [--keep_ins_bams] [--detail_out DETAIL_OUT] [--unmapped]
                  [--te] [--usecachedLAST] [--uuid_list UUID_LIST]
                  [--tmpdir TMPDIR]

Resolve insertions from TEbreak data

optional arguments:
  -h, --help            show this help message and exit
  -p PICKLE, --pickle PICKLE
                        pickle file output from tebreak.py
  -t PROCESSES, --processes PROCESSES
                        split work across multiple processes
  -i INSLIB_FASTA, --inslib_fasta INSLIB_FASTA
                        reference for insertions (not genome)
  -m FILTER_BED, --filter_bed FILTER_BED
                        BED file of regions to mask
  -v, --verbose         output status information
  --max_bam_count MAX_BAM_COUNT
                        skip sites with more than this number of BAMs (default
                        = no limit)
  --min_ins_match MIN_INS_MATCH
                        minumum match to insertion library
  --minmatch MINMATCH   minimum match to reference genome
  --annotation_tabix ANNOTATION_TABIX
                        can be comma-delimited list
  --refoutdir REFOUTDIR
                        output directory for generating tebreak references
                        (default=tebreak_refs)
  --skip_align          skip re-alignment of discordant ends to ref insertion
  --use_rg              use RG instead of BAM filename for samples
  --keep_all_tmp_bams   leave ALL temporary BAMs (warning: lots of files!)
  --keep_ins_bams       insertion-specific BAMs will be kept in --refoutdir
  --detail_out DETAIL_OUT
                        file to write detailed output
  --unmapped            report insertions that do not match insertion library
  --te                  set if insertion library is transposons
  --usecachedLAST       try to used cached LAST db, if found
  --uuid_list UUID_LIST
                        limit resolution to UUIDs in first column of input
                        list (can be tabular output from previous resolve.py
                        run)
  --tmpdir TMPDIR       directory for temporary files
```

4.2 Description

This script is the second step in insertion analysis via TEBreak, it is separated from the initial insertion discovery script (tebreak.py) to facilitate running multiple different analyses on the same set of putative insertion sites (e.g. detecting transposable elements, viral insertions, processed transcript insertions, and novel sequence insertions from the same WGS data).

4.3 Input

Insertion call input (-p/–pickle) This is the 'pickle' containing information about putative insertion sites derived from tebreak.py (filename specified by –pickle).

Reference genome (-i/–inslib A FASTA file containing template insertion sequences (e.g. reference transposable elements, viral sequences, mRNAs, etc.). For transposable elements, sequence superfamilies and subfamilies can be specified by separating with a colon (:) as follows:

```
>ALU:AluYa5
GGCCGGGCGCGGTGGCTCACGCCTGTAATCCCAGCACTTTGGGAGGCCGAGGCGGGCGGATCACGAGGTCAGGAGATCG
AGACCATCCCGGCTAAAACGGTGAAACCCCGTCTCTACTAAAAATACAAAAAATTAGCCGGGCGTAGTGGCGGGCGCCT
GTAGTCCCAGCTACTTGGGAGGCTGAGGCAGGAGAATGGCGTGAACCCGGGAGGCGGAGCTTGCAGTGAGCCGAGATCC
CGCCACTGCACTCCAGCCTGGGCGACAGAGCGAGACTCCGTCTCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Detailed human-readable output (–detail_out) This is a file containing detailed information about consensus reads, aligned segments, and statistics for each putative insertion site detected. Note that this is done with minimal filtering, so these should not be used blindly. Default filename is resolve.out.

Additional options

- -t/–threads : Split work across multiple processes.
- -m/–filter_bed : BED file of regions to mask (will not be output)
- -v/–verbose : Output status information.
- –max_bam_count : do not analyse insertions represented by more than this number of BAMs (default = no filter)

- –min_ins_match : minimum percent match to insertion library
- –minmatch : minimum percent match to reference genome
- –annotation_tabix : tabix-indexed file, entries overlapping insertions will be annotated in output
- –refoutdir : non-temporary output directory for various references generated by resolve.py. This includes LAST references for the insertion library and insertion-specific BAMs if –keep_ins_bams is enabled.
- –skip_align : skip insertion-specific realignment of split/discordant reads (not recommended on first pass)
- –use_rg : use the readgroup name instead of the BAM name to count and annotate samples
- –keep_all_tmp_bams : retain all temporary BAMs in temporary directory (warning: this can easily be in excess of 100000 files for WGS data and may lead to unhappy filesystems).
- –unmapped : also report insertions that do not match the insertion library
- –te : enable additional filtering specific to transposable element insertions
- –usecachedLAST : useful if -i/–inslib FASTA is large, you can use a pre-built LAST reference (in –refoutdir) e.g. if it was generated on a previous run.
- –uuid_list : limit analysis to set of UUIDs in the first column of specified file (generally, this is the table output by a previous run of resolve.py) - this is useful for changing annotations, altering parameters, debugging, etc.
- –tmpdir : directory for temporary files (default is /tmp)

4.4 Output

A table (tab-delimited) is output (with a header) is written to STDOUT, so I would recommend redirecting stdout of resolve.py to a file. The output contains the following columns:

- Chromosome : Chromosome
- Left_Extreme : Coordinate of the left-most reference base mapped to a read supporting the insertion
- Right_Extreme : Coordinate of the right-most reference base mapped to a read supporting the insertion
- 5_Prime_End : Breakend corresponding to the 5' end of the inserted sequence (where 5' is can be defined e.g. for transposable elements)
- 3_Prime_End : Breakend corresponding to the 3' end of the inserted sequence (where 3' is can be defined e.g. for transposable elements)
- Superfamily : Superfamily of insertions of superfamily is defined (see option -i/–inslib in resolve.py)

- Subfamily : Subfamily of insertions of subfamily is defined (see option -i/–inslib in resolve.py)
- TE_Align_Start : alignment start position within inserted sequence relative to reference
- TE_Align_End : alignment end position within inserted sequence relative to reference
- Orient_5p : Orientation derived from the 5' end of the insertion (if 5' is defined), assuming insertion reference is read 5' to 3'
- Orient_3p : Orientation derived from the 3' end of the insertion (if 3' is defined), assuming insertion reference is read 5' to 3'
- Inversion : If Orient_5p and Orient_3p disagree, there may be an inversion in the inserted sequence relative to the reference sequence for the insertion
- 5p_Elt_Match : Fraction of bases matched to reference for inserted sequence on insertion segment of 5' supporting contig
- 3p_Elt_Match : Fraction of bases matched to reference for inserted sequence on insertion segment of 3' supporting contig
- 5p_Genome_Match : Fraction of bases matched to reference genome on genomic segment of 5' supporting contig
- 3p_Genome_Match : Fraction of bases matched to reference genome on genomic segment of 3' supporting contig
- Split_reads_5prime : Number of split (clipped) reads supporting 5' end of the insertion
- Split_reads_3prime : Number of split (clipped) reads supporting 5' end of the insertion
- Remapped_Discordant : Number of discordant read ends re-mappable to insertion reference sequence
- Remapped_Splitreads : Number of split reads re-mappable to insertion reference sequence
- 5p_Cons_Len : Length of 5' consensus sequence (Column Consensus_5p)
- 3p_Cons_Len : Length of 3' consensus sequence (Column Consensus_3p)
- 5p_Improved : Whether the 5' consensus sequence was able to be improved through local assembly in tebreak.py
- 3p_Improved : Whether the 3' consensus sequence was able to be improved through local assembly in tebreak.py
- TSD_3prime : Target site duplication (if present) based on 5' overlap on consensus sequence
- TSD_5prime : Target site duplication (if present) based on 3' overlap on consensus sequence (can be different from 5' end, but in many cases it is advisable to filter out putative insertions that disagree on TSDs)
- Sample_count : Number of samples (based on BAM files or readgroups, depending on –use_rg option in resolve.py)

- Sample_support : Per-sample split read count supporting insertion presence (Sample|Count)
- Consensus_5p : Consensus sequence for 5' supporting contig
- Consensus_3p : Consensus sequence for 3' supporting contig

Additional columns may be present depending on annotation options.

5 Filtering

5.1 Pre-filtering

In some cases it is not necessary to analyse the entire genome. Regions of interest in BED format may be input to tebreak.py via the -i/–inverval_bed option.

- If paired-end reads are present, it is possible to pre-select regions where insertions may be indicated by discordant paired-end mappings. This can be accomplished using the script 'discoreads.py' which is in the scripts directory of the TEBreak distribution.
- For whole genome sequencing (WGS) data, it is only the soft-clipped reads and reads in discordant pairs that are informative; the rest of the mapped reads can be discarded. This can be accomplished using the script 'reduce_bam.py', located in the scripts directory.
- For sequence derived from targeted sequencing methods (e.g. whole exome sequencing, RC-seq), it may be useful to only analyse regions covered to at least a certain read depth. This can be accomplished using the script 'covered_segments.py' in the scripts directory.

5.2 Post-filtering

Generally additional filtering is required on the tabular output to achieve the desired level of accuracy. For example, the positions of transposable element insertions should be filtered against the locations of transposable element insertions in the reference genome used for input to tebreak.py.