SomaticCall manual

SomaticCall is a program that finds single-base differences (substitutions) between sequence data from tumor and matched normal samples. It is designed to be highly stringent, so as to achieve a low false positive rate. It takes as input a BAM file for each sample, and produces as output a list of differences (somatic mutations).

CAVEATS
(1) There are some issues with the naming of chromosomes in report files. This code was originally tested on hg18 and relied on some naming conventions for fasta records in it. We have attempted to eliminate this problem but have not tested the changes sufficiently to be sure if they work generally.

1. Installation

(a) download the software

(b) make sure you have a recent version of GCC and addr2line

(c) you need GMP with C++ support

(d) you also need 'samtools' in your path

(e) Type './configure' then 'make'.

(f) make sure that the several executables created by make are all in your path

(g) If you have trouble, please see general help instructions at http://www.broadinstitute.org/science/programs/genome-biology/crd.

2. Generating BAM files as input

Currently we do not provide instructions for creating BAM files. However, please note the following:

(a) It is absolutely critical that duplicates be flagged correctly. If you find that SomaticCall produces false positives, incorrect flagging of duplicates is the most likely cause.

(b) The BAM file should include unplaced reads.

(c) The quality scores in the BAM file should have been recalibrated.

(d) We anticipate that the SomaticCall algorithm will not work well on data having many indel errors.

(e) For the Illumina platform, we recommend using all reads, not just purity-filtered (PF) reads.

3. Generating binary reference files

To generate the REFHEAD.fastb and REFHEAD.lookup files, run
MakeLookupTable SOURCE=g OUT_HEAD=REFHEAD
where
* g is the fasta file for the genome reference
* REFHEAD is the output "head"; will create REFHEAD.fastb and REFHEAD.lookup.

4. Call mutations

(a) Index the BAM files using 'samtools index'.

(d) Invoke `SomaticCall` with the following options

`TUMOR_BAM=` the name of your tumor BAM file (ends in .bam)

`NORMAL_BAM=` the name of your normal BAM file (ends in .bam)

`REFHEAD=` (see previous section; there must exist files REFHEAD.fasta, REFHEAD.fastb, and REFHEAD.lookup)

`DBSNP=` name of file containing one line for each known SNP, where each line has two fields: the chromosome name, and the one-based position of the SNP on the chromosome; chromosome names must match those in the fasta file for the reference

`MAX_THREADS=` maximum number of threads to run in parallel

`OUT=` the name of the directory where the results should be placed (this directory should not be shared with any other process)

5. Find the mutations

They are in the file `OUT/mutation_reports`. Column one provides zero-based coordinates.

6. Under the hood: description of the algorithm and intermediate files

(NOTE: This isn't quite right anymore. The last file created is mutation_reports4.)

(A) Handling of alignments. They are discarded if they are marked as duplicates or assigned mapping quality score 0. The mapping quality score is not otherwise used.

(B) Core statistical computation. We describe a test that is repeated several times in the process. Given aligned reads for the tumor and normal, the code SomaticMutation in SomaticCallTools.cc tests for a likely somatic mutation according to the following criteria:

(i) either an adjusted quality score sum in the tumor for the mutant base must be at least 100 or the LOD score for mutant:ref + mutant:mutant vs ref:ref must be at least 6.3;

(ii) the quality score sum for the mutant base in the normal must be < 50 and the LOD score for ref:ref vs mutant:ref + mutant:mutant must be at least 2.3.

- This test is applied several times as the alignments are modified in the course of the program. Once a putative mutation fails the test, it is not considered again.
- There is also a pretest which checks that the adjusted quality score sum in the tumor is at least 60, but this is intended only as an optimization.
- Note that the adjusted quality score sum test of (i) does not actually affect the final output, but is included for experimental purposes.

(C) Overall steps in the process.

(i) First, dbSNP position are excluded, and a list of putative somatic mutations is obtained by applying SomaticMutation (yielding mutation_reports0).

(ii) Then we require that a graph assembly of the locus has no cycles and that all paths across it have the same length (yielding mutation_reports1).

(iii) All reads supporting the mutant allele in the tumor are then realigned at high stringency. We seed on 12-mers of frequency up to 1000, allow up to 4 errors (including indels of size up to 2), and require that the next best placement has more than 6 more errors. Then we call SomaticMutation again (yielding mutation_reports2).

(iv) Next we eliminate mutations for which the alternative base is supported by only one read start position (yielding mutation_reports3).

(v) Then we search the entire set of reads from the normal sample with the goal of finding more reads at each putative mutation.

(a) To do this we first examine the given tumor alignments to find 21-mers from the tumor data that are centered at the mutation. Find the most frequent 3 such 21-mers. If there are no such 21-mers, reject the locus. (At this point we create mutation_reports4.)

(b) For each of the two 20-mers in the above 21-mers, find every instance in the normal reads.

(c) For each such instance, align the read to the reference at the locus of the somatic mutation, allowing up to 6 mismatches and indels.

(d) Align the read to the whole genome using 12-mers, not allowing indels. If an alignment is found that is nearly as good or better than the local placement, discard the read.
(e) Return the local placements of the surviving reads.
(f) Add in the additional normal reads and recall mutations.

This process yields mutation_reports5.

(vi) Finally we filter for tumor score >= 6.3, normal_score 2.3, yielding mutation_reports5f. We also create a link mutation_reports that points to this. Each entry in these files has a reference to report_dir, a directory that contains files t.visual and n.visual, that can be viewed with less -r to show the reads that support a given mutation.