SomaticSeq Documentation

Li Tai Fang

September 5, 2017

1 Introduction

SomaticSeq is a flexible post-somatic-mutation-calling workflow for improved accuracy. We have incorporated multiple somatic mutation caller(s) to obtain a combined call set, and then it uses machine learning to distinguish true mutations from false positives from that call set. We have incorporated the following somatic mutation caller: MuTect/Indelocator, MuTect2, VarScan2, JointSNVMix, SomaticSniper, VarDict, MuSE, LoFreq, and Scalpel. You may incorporate some or all of those callers into your own pipeline with SomaticSeq.

The manuscript, An ensemble approach to accurately detect somatic mutations using SomaticSeq, is published in Genome Biology 2015, 16:197. The SomaticSeq project is located at *http://bioinform.github.io/somaticseq/*. The data described in the manuscript is also described at *http://bioinform.github.io/somaticseq/data.html*. There have been some major improvements since the publication.

SomaticSeq.Wrapper.sh is a bash script that calls a series of scripts to combine the output of the somatic mutation caller(s), after the somatic mutation callers are run. Then, depending on what R scripts are fed to SomaticSeq.Wrapper.sh, it will either 1) train the call set into a classifier, 2) predict high-confidence somatic mutations from the call set based on a pre-defined classifier, or 3) simply label the calls (i.e., PASS, LowQual, or REJECT) based on majority vote of the tools.

1.1 Dependencies

- Python 3, plus regex, pysam, numpy, and scipy libraries. All the .py scripts are written in Python 3.
- R, plus the ada package in R.
- BEDTools (if there is an inclusion and/or an exclusion region file)
- Optional: Free GATK version 3. You can use GATK CombineVariants to merge VCF files if the jar file is provided to `--gatk`. Otherwise, the sorting is done by the vcfsort.pl script written by German Gaston Leparc.
- Optional: dbSNP and COSMIC files in VCF format (if you want to use these features as a part of the training).
- At least one of MuTect/Indelocator/MuTect2, VarScan2, JointSNVMix2, SomaticSniper, VarDict, MuSE, LoFreq, Scalpel, and/or Strelka2. Those are the tools we have incorporated in SomaticSeq. If there are other somatic tools that may be good addition to our list, please make the suggestion to us.

1.2 Docker repos

SomaticSeq and (almost) all the somatic mutation callers we have incorporated are now dockerized. The exception is VarScan2 because we do not have distribution rights.

- SomaticSeq is dockerized at https://hub.docker.com/r/lethalfang/somaticseq/.
- MuTect2 (tested with GATK:4.beta.3): https://hub.docker.com/r/broadinstitute/gatk
- VarScan2 (untested): https://hub.docker.com/r/djordjeklisic/sbg-varscan2/
- JointSNVMix2: https://hub.docker.com/r/lethalfang/jointsnvmix2/
- SomaticSniper: https://hub.docker.com/r/lethalfang/somaticsniper/
- VarDict: https://hub.docker.com/r/lethalfang/vardictjava/
- MuSE: https://hub.docker.com/r/marghoob/muse/
- LoFreq: https://hub.docker.com/r/marghoob/lofreq/
- Scalpel: https://hub.docker.com/r/lethalfang/scalpel/
- Strelka2: https://hub.docker.com/r/lethalfang/strelka/

2 To use SomaticSeq.Wrapper.sh

The SomaticSeq.Wrapper.sh is a wrapper script that calls a series of programs and procedures **after** you have run your individual somatic mutation callers. In the next section, we will describe the workflow in more detail, so you may not be dependent on this wrapper script. You can either modify this wrapper script or create your own workflow.

2.1 To train data set into a classifier

To create a trained classifier, ground truth files are required for the data sets. There is also an option to include a list of regions to include and/or exclude. The exclusion or inclusion regions can be VCF or BED files. An inclusion region may be subset of the call sets where you have validated their true/false mutation status, so that only those regions will be used for training. An exclusion region can be regions where the "truth" is ambigious.

All the output VCF files from individual callers are optional. Those VCF files can be bgzipped if they have .vcf.gz extensions. It is imperative that the parameters used for the training and prediction are identical.

```
# For training, truth file and the correct R script are required.

SomaticSeq.Wrapper.sh \
--mutect              MuTect/variants.snp.vcf \
--mutect2             MuTect2/variants.vcf \
--indelocator         Indelocator/variants.indel.vcf \
--varscan-snv         VarScan2/variants.snp.vcf \
--varscan-indel       VarScan2/variants.indel.vcf \
--jsm                 JointSNVMix2/variants.snp.vcf \
--sniper              SomaticSniper/variants.snp.vcf \
--vardict             VarDict/variants.vcf \
--muse                MuSE/variants.snp.vcf \
--lofreq-snv          LoFreq/variants.snp.vcf \
--lofreq-indel        LoFreq/variants.indel.vcf \
```

```
--scalpel            Scalpel/variants.indel.vcf \
--strelka-snv        Strelka/variants.snv.vcf \
--strelka-indel      Strelka/variants.indel.vcf \
--normal-bam         matched_normal.bam \
--tumor-bam          tumor.bam \
--ada-r-script       ada_model_builder.R \
--genome-reference   human_b37.fasta \
--cosmic             cosmic.b37.v71.vcf \
--dbsnp              dbSNP.b37.v141.vcf \
--gatk               $PATH/TO/GenomeAnalysisTK.jar \
--exclusion-region   ignore.bed \
--inclusion-region   validated.bed
--truth-snv          truth.snp.vcf \
--truth-indel        truth.indel.vcf \
--output-dir         $OUTPUT_DIR
```

SomaticSeq.Wrapper.sh supports any combination of the somatic mutation callers we have incorporated into the workflow. SomaticSeq will run based on the output VCFs you have provided. It will train for SNV and/or INDEL if you provide the truth.snp.vcf and/or truth.indel.vcf file(s) as well as the proper R script (ada_model_builder.R). Otherwise, it will fall back to the simple caller consensus mode.

2.2 To predict somatic mutation based on trained classifiers

Make sure the classifiers and the proper R script (ada_model_predictor.R) are supplied, Without either of them, it will fall back to the simple caller consensus mode.

```
# The *.RData files are trained classifier from the training mode.
SomaticSeq.Wrapper.sh \
--mutect             MuTect/variants.snp.vcf \
--mutect2            MuTect2/variants.vcf \
--indelocator        Indelocator/variants.indel.vcf \
--varscan-snv        VarScan2/variants.snp.vcf \
--varscan-indel      VarScan2/variants.indel.vcf \
--jsm                JointSNVMix2/variants.snp.vcf \
--sniper             SomaticSniper/variants.snp.vcf \
--vardict            VarDict/variants.vcf \
--muse               MuSE/variants.snp.vcf \
--lofreq-snv         LoFreq/variants.snp.vcf \
--lofreq-indel       LoFreq/variants.indel.vcf \
--scalpel            Scalpel/variants.indel.vcf \
--strelka-snv        Strelka/variants.snv.vcf \
--strelka-indel      Strelka/variants.indel.vcf \
--normal-bam         matched_normal.bam \
--tumor-bam          tumor.bam \
--ada-r-script       ada_model_predictor.R \
--genome-reference   human_b37.fasta \
--cosmic             cosmic.b37.v71.vcf \
--dbsnp              dbSNP.b37.v141.vcf \
--snpeff-dir         $PATH/TO/DIR/snpSift \
--gatk               $PATH/TO/GenomeAnalysisTK.jar \
--exclusion-region   ignore.bed \
--inclusion-region   validated.bed
--classifier-snv     sSNV.Classifier.RData \
--classifier-indel   sINDEL.Classifier.RData \
--pass-threshold     0.5 \
--lowqual-threshold  0.1 \
--output-dir         $OUTPUT_DIR
```

If both MuTect2 and MuTect/Indelocator VCF files are provided, the script is written such that it will use MuTect2 and ignore MuTect.

2.3 To classify based on simple majority vote

Same as the command previously, but not including the R script or the ground truth files. Without those information, SomaticSeq will fall back into a simple majority vote.

3 The step-by-step SomaticSeq Workflow

We'll describe the workflow here, so you may modify the workflow and/or create your own workflow instead of using the wrapper script described previously.

3.1 Combine the call sets

We use GATK CombineVariants to combine the VCF files fkeep-intermediates:rom different callers, although it does not matter what tools are used to merge VCF files. We GATK CombineVariants because it's quite fast. To make them compatible with GATK, the VCF files are modified.

A simple alternative method is to use GNU sort and uniq in Linux to list all the unique first-5-column (i.e., CHROM, POS, ID, REF, and ALT) from all the VCF files, and then fill the remaining required VCF columns with whatever string and sort it according to the reference. That VCF file will do the job just fine.

For our GATK CombineVariants procedure:

1. Modify MuTect and/or Indelocator output VCF files. Since MuTect's output VCF do not always put the tumor and normal samples in the same columns, the script will determine that information based on either the BAM files (the header has sample name information), or based on the sample information that you tell it, and then determine which column belongs to the normal, and which column belongs to the tumor.

```
# Modify MuTect and Indelocator's output VCF based on BAM files
modify_MuTect.py -infile input.vcf -outfile output.vcf -nbam normal.bam -tbam
    tumor.bam

# Based on the sample name you supply:
modify_MuTect.py -infile input.vcf -outfile output.vcf -nsm NormalSampleName -
    tsm TumorSampleName
```

2. For MuTect2, this script will split multi-allelic records into one variant per line in the VCF file. This is to make thing easier for the SSeq_merged.vcf2tsv.py script later.

```
# Based on the sample name you supply:
modify_MuTect2.py -infile MuTect2.Filtered.vcf -snv mutect.snp.vcf -indel
    mutect.indel.vcf
```

3. Modify VarScan's output VCF files to be rigorously concordant to VCF format standard, and to attach the tag 'VarScan2' to somatic calls.

```
# Do it for both the SNV and indel
modify_VJSD.py -method VarScan2 -infile input.vcf -outfile output.vcf
```

4. JointSNVMix2 does not output VCF files. In our own workflow, we have already converted its text file into a basic VCF file with an 2 awk one-liners, which you may see in the Run_5_callers directory, which are:

```
# To avoid text files in the order of terabytes, this awk one-liner keeps
    entries where the reference is not "N", and the somatic probabilities are
    at least 0.95.
awk -F "\t" 'NR!=1 && $4!="N" && $10+$11>=0.95'

# This awk one-liner converts the text file into a basic VCF file
awk -F "\t" '{print $1 "\t" $2 "\t.\t" $3 "\t" $4 "\t.\t.\tAAAB=" $10 ";AABB="
     $11 "\tRD:AD\t" $5 ":" $6 "\t" $7 ":" $8}'

## The actual commands we've used in our workflow:
echo -e '##fileformat=VCFv4.1' > unsorted.vcf
echo -e '##INFO=<ID=AAAB,Number=1,Type=Float,Description="Probability of Joint
     Genotype AA in Normal and AB in Tumor">' >> unsorted.vcf
echo -e '##INFO=<ID=AABB,Number=1,Type=Float,Description="Probability of Joint
     Genotype AA in Normal and BB in Tumor">' >> unsorted.vcf
echo -e '##FORMAT=<ID=RD,Number=1,Type=Integer,Description="Depth of reference
    -supporting bases (reads1)">' >> unsorted.vcf
echo -e '##FORMAT=<ID=AD,Number=1,Type=Integer,Description="Depth of variant-
    supporting bases (reads2)">' >> unsorted.vcf
echo -e '#CHROM\tPOS\tID\tREF\tALT\tQUAL\tFILTER\tINFO\tFORMAT\tNORMAL\tTUMOR'
     >> unsorted.vcf

python $PATH/TO/jsm.py classify joint_snv_mix_two genome.GRCh37.fa normal.bam
    tumor.bam trained.parameter.cfg /dev/stdout |\
awk -F "\t" 'NR!=1 && $4!="N" && $10+$11>=0.95' |\
awk -F "\t" '{print $1 "\t" $2 "\t.\t" $3 "\t" $4 "\t.\t.\tAAAB=" $10 ";AABB="
     $11 "\tRD:AD\t" $5 ":" $6 "\t" $7 ":" $8}' >> unsorted.vcf
```

After that, you'll also want to sort the VCF file. Now, to modify that basic VCF into something that will be compatible with other VCF files under GATK CombineVariants:

```
modify_VJSD.py -method JointSNVMix2 -infile input.vcf -outfile output.vcf
```

5. Modify SomaticSniper's output:

```
modify_VJSD.py -method SomaticSniper -infile input.vcf -outfile output.vcf
```

6. VarDict has both SNV and indel, plus some other variants in the same VCF file. Our script will create two files, one for SNV and one for indel, while everything else is ignored for now. By default, LikelySomatic/StrongSomatic and PASS calls will be labeled VarDict. However, in our SomaticSeq paper, based on our experience in DREAM Challenge, we implemented two custom filters to relax the VarDict tagging criteria.

```
# Default VarDict tagging criteria, only PASS (and Likely or Strong Somatic):
modify_VJSD.py -method VarDict -infile intput.vcf -outfile output.vcf

# When running VarDict, if var2vcf_paired.pl is used to generate the VCF file,
    you may relax the tagging criteria with -filter paired
modify_VJSD.py -method VarDict -infile intput.vcf -outfile output.vcf -filter
    paired

# When running VarDict, if var2vcf_somatic.pl is used to generate the VCF file
    , you may relax the tagging criteria with -filter somatic
modify_VJSD.py -method VarDict -infile intput.vcf -outfile output.vcf -filter
    somatic
```

In the SomaticSeq paper, -filter somatic was used because var2vcf_somatic.pl was used to generate VarDict's VCF files. In the SomaticSeq.Wrapper.sh script, however, -filter paired is used because VarDict authors have since recommended var2vcf_paired.pl script to create the VCF files. While there are some differences (different stringencies in some filters) in what VarDict labels as PASS between the somatic.pl and paired.pl scripts, the difference is miniscule after applying our custom filter (which relaxes the filter, resulting in a difference about 5 calls out of 15,000).

The output files will be snp.output.vcf and indel.output.vcf.

7. MuSE was not a part of our analysis in the SomaticSeq paper. We have implemented it later.

```
modify_VJSD.py -method MuSE -infile input.vcf -outfile output.vcf
```

8. LoFreq and Scalpel do not require modification. LoFreq has no sample columns anyway.

9. Add "GT" field to sample columns to make it compatible with GATK CombineVariants.

```
modify_Strelka.py -infile somatic.snvs.vcf.gz -outfile stralka.snv.vcf
```

10. Finally, with the VCF files modified, you may combine them with GATK CombineVariants: one for SNV and one for indel separately. There is no particular reason to use GATK CombineVariants. Other combiners should also work. The only useful thing here is to combine the calls and GATK CombineVariants does it well and pretty fast.

```
# Combine the VCF files for SNV. Any or all of the VCF files may be present.
# -nt 6 means to use 6 threads in parallel
java -jar $PATH/TO/GenomeAnalysisTK.jar -T CombineVariants -R genome.GRCh37.fa
    -nt 6 --setKey null --genotypemergeoption UNSORTED -V mutect.vcf -V
    varscan.snp.vcf -V jointsnvmix.vcf -V snp.vardict.vcf -V muse.vcf --out
    CombineVariants.snp.vcf
java -jar $PATH/TO/GenomeAnalysisTK.jar -T CombineVariants -R genome.GRCh37.fa
    -nt 6 --setKey null --genotypemergeoption UNSORTED -V indelocator.vcf -V
    varscan.snp.vcf -V indel.vardict.vcf --out CombineVariants.indel.vcf
```

3.2 For model training: process and annotate the VCF files (union of call sets)

This step may be needed for model training. The workflow in SomaticSeq.Wrapper.sh allows for inclusion and exclusion region. An inclusion region means we will only use calls inside these regions. An exclusion region means we do not care about calls inside this region. DREAM Challenge had exclusion regions, e.g., blacklisted regions, etc.

```
# In the DREAM_Stage_3 directory, we have included an exclusion region BED file as an example
# This command uses BEDtools to rid of all calls in the exclusion region
intersectBed -header -a BINA_somatic.snp.vcf   -b ignore.bed -v > somatic.snp.processed.vcf
intersectBed -header -a BINA_somatic.indel.vcf -b ignore.bed -v > somatic.indel.processed.vcf

# Alternatively (or both), this command uses BEDtools to keep only calls in the inclusion region
intersectBed -header -a BINA_somatic.snp.vcf   -b inclusion.bed > somatic.snp.processed.vcf
intersectBed -header -a BINA_somatic.indel.vcf -b inclusion.bed > somatic.indel.processed.vcf
```

3.3 Convert the VCF file, annotated or otherwise, into a tab separated file

This script works for all VCF files. It extracts information from BAM files as well as some VCF files created by the individual callers. If the ground truth VCF file is included, a called variant will be annotated as a true positive, and everything will be annotated as a false positive.

```
# SNV
SSeq_merged.vcf2tsv.py -ref genome.GRCh37.fa -myvcf somatic.snp.processed.vcf -truth Ground.truth.snp.vcf -mutect MuTect/variants.snp.vcf.gz -varscan VarScan2/variants.snp.vcf -jsm JSM2/variants.vcf -sniper SomaticSniper/variants.vcf -vardict VarDict/snp.variants.vcf -muse MuSE/variants.vcf -lofreq LoFreq/variants.snp.vcf -strelka Strelka/variants.snp.vcf -dedup -tbam tumor.bam -nbam normal.bam -outfile Ensemble.sSNV.tsv
```

That was for SNV, and indel is almost the same thing. After version 2.1, we have replaced all information from SAMtools and HaplotypeCaller with information directly from the BAM files. The accuracy differences are negligible with significant improvement in usability and resource requirement.

```
# INDEL:
SSeq_merged.vcf2tsv.py -ref genome.GRCh37.fa -myvcf somatic.indel.processed.vcf -truth Ground.truth.indel.vcf -varscan VarScan2/variants.snp.vcf -vardict VarDict/indel.variants.vcf -lofreq LoFreq/variants.indel.vcf -scalpel Scalpel/variants.indel.vcf -strelka Strelka/variants.indel.vcf -tbam tumor.bam -nbam normal.bam -dedup -outfile Ensemble.sINDEL.tsv
```

At the end of this, Ensemble.sSNV.tsv and Ensemble.sINDEL.tsv are created.

All the options for SSeq_merged.vcf2tsv.py are listed here. They can also be displayed by running SSeq_merged.vcf2tsv.py --help.

```
  -myvcf        Input VCF file of the merged calls [REQUIRED]
  -ref          Genome reference fa/fasta file [REQUIRED]
  -nbam         BAM file of the matched normal sample [REQUIRED]
  -tbam         BAM file of the tumor sample [REQUIRED]
  -ref          Genome reference fa/fastq file [REQUIRED]
  -truth        Ground truth VCF file. Every other position is a False Positive.
  -dbsnp        dbSNP VCF file
  -cosmic       COSMIC VCF file
  -mutect       VCF file from either MuTect2, MuTect, or Indelocator
  -sniper       VCF file from SomaticSniper
  -varscan      VCF file from VarScan2
  -jsm          VCF file from Bina's workflow that contains JointSNVMix2
  -vardict      VCF file that contains only SNV or only INDEL from VarDict
  -muse         VCF file from MuSE
  -lofreq       VCF file from LoFreq
  -scalpel      VCF file from Scalpel
  -strelka      VCF file from Strelka
  -dedup        A flag to consider only primary reads
  -minMQ        Minimum mapping quality for reads to be considered (Default = 1)
  -minBQ        Minimum base quality for reads to be considered (Default = 5)
  -mincaller    Minimum number of caller classification for a call to be considered
    (Use 0.5 to consider some LowQual calls. Default = 0).
  -scale        The options are phred, fraction, or None, to convert numbers to
    Phred scale or fractional scale. (default = None, i.e., no conversion)
  -outfile      Output TSV file name
```

Note: Do not worry if Python throws a warning like this.

```
RuntimeWarning: invalid value encountered in double_scalars
  z = (s - expected) / np.sqrt(n1*n2*(n1+n2+1)/12.0)
```

This is to tell you that scipy was attempting some statistical test with empty data. That's usually due to the fact that normal BAM file has no variant reads at that given position. That is why lots of values are NaN for the normal.

3.4 Model Training or Mutation Prediction

You can use Ensemble.sSNV.tsv and Ensemble.sINDEL.tsv files either for model training (provided that their mutation status is annotated with 0 or 1) or mutation prediction. This is done with stochastic boosting algorithm we have implemented in R.

Model training:

```
# Training:
r_script/ada_model_builder.R Ensemble.sSNV.tsv
r_script/ada_model_builder.R Ensemble.sINDEL.tsv
```

Ensemble.sSNV.tsv.Classifier.RData and Ensemble.sINDEL.tsv.Classifier.RData will be created from model training.

Mutation prediction:

```
# Mutation prediction:
r_script/ada_model_predictor.R Ensemble.sSNV.tsv.Classifier.RData    Ensemble.sSNV.
    tsv    Trained.sSNV.tsv
```

```
r_script/ada_model_predictor.R Ensemble.sINDEL.tsv.Classifier.RData Ensemble.sINDEL.tsv Trained.sINDEL.tsv
```

After mutation prediction, if you feel like it, you may convert Trained.sSNV.tsv and Trained.sINDEL.tsv into VCF files. Use -tools to list ONLY the individual tools used to have appropriately annotated VCF files. Accepted tools are CGA (for MuTect/Indelocator), VarScan2, JointSNVMix2, SomaticSniper, VarDict, MuSE, LoFreq, and/or Scalpel. To list a tool without having run it, the VCF will be annotated as if the tool was run but did not identify that position as a somatic variant.

```
# Probability above 0.7 labeled PASS (-pass 0.7), and between 0.1 and 0.7 labeled LowQual (-low 0.1):
# Use -all to include REJECT calls in the VCF file
# Use -phred to convert probability values (between 0 to 1) into Phred scale in the QUAL column in the VCF file

SSeq_tsv2vcf.py -tsv Trained.sSNV.tsv   -vcf Trained.sSNV.vcf   -pass 0.7 -low 0.1 -tools CGA VarScan2 JointSNVMix2 SomaticSniper VarDict MuSE LoFreq Strelka -all -phred

SSeq_tsv2vcf.py -tsv Trained.sINDEL.tsv -vcf Trained.sINDEL.vcf -pass 0.7 -low 0.1 -tools CGA VarScan2 VarDict LoFreq Scalpel Strelka -all -phred
```

4 To run the dockerized somatic mutation callers

For your convenience, we have created a couple of scripts that can generate run script for the dockerized somatic mutation callers.

4.1 Location

- somaticseq/utilities/dockered_pipelines/

4.2 Requirements

- Have internet connection, and able to pull and run docker images from docker.io
- Have cluster management system such as Sun Grid Engine, so that the "qsub" command is valid

4.3 Example commands

For single-threaded jobs, best suited for whole exome sequencing or less.

```
# Example command to submit the run scripts for each of the following somatic mutation callers
utilities/dockered_pipelines/singleThread/submit_callers_singleThread.sh \
--normal-bam       /ABSOLUTE/PATH/TO/normal_sample.bam \
--tumor-bam        /ABSOLUTE/PATH/TO/tumor_sample.bam \
--human-reference  /ABSOLUTE/PATH/TO/GRCh38.fa \
--output-dir       /ABSOLUTE/PATH/TO/RESULTS \
--selector         /ABSOLUTE/PATH/TO/Exome_Capture.GRCh38.bed \
--dbsnp            /ABSOLUTE/PATH/TO/dbSNP.GRCh38.vcf \
```

```
--action          qsub \
--mutect2 --jointsnvmix2 --somaticsniper --vardict --muse --lofreq --scalpel --
    strelka --somaticseq
```

For multi-threaded jobs, best suited for whole genome sequencing.

```
# Submitting mutation caller jobs by splitting each job into 36 even regions.
utilities/dockered_pipelines/multiThreads/submit_callers_multiThreads.sh \
--normal-bam       /ABSOLUTE/PATH/TO/normal_sample.bam \
--tumor-bam        /ABSOLUTE/PATH/TO/tumor_sample.bam \
--human-reference  /ABSOLUTE/PATH/TO/GRCh38.fa \
--output-dir       /ABSOLUTE/PATH/TO/RESULTS \
--dbsnp            /ABSOLUTE/PATH/TO/dbSNP.GRCh38.vcf \
--threads          36 \
--action           qsub \
--mutect2 --somaticsniper --vardict --muse --lofreq --scalpel --strelka --
    somaticseq
```

4.3.1 What does the single-threaded command do

- For each flag such as --mutect2, --jointsnvmix2, ..., --strelka, a run script ending with .cmd will be created in /ABSOLUTE/PATH/TO/RESULTS/logs. By default, these .cmd scripts will only be created, and their file path will be printed on screen. However, if you do "--action qsub", then these scripts will be submitted via the qsub command. The default action is "echo."
  - Each of these .cmd script correspond to a mutation caller you specified. They all use docker images.
  - We may improve their functionalities in the future to allow more tunable parameters. For the initial releases, POC and reproducibility take precedence.
- If you do "--somaticseq," the somaticseq script will be created in /ABSOLUTE/PATH/-TO/RESULTS/SomaticSeq/logs. However, it will not be submitted until you manually do so after each of these mutation callers is finished running.
  - In the future, we may create more sophisticated solution that will automatically solves these dependencies. For the initial release, we'll focus on stability and reproducibility.
- Due to the way those run scripts are written, the Sun Grid Engine's standard error log will record the time the task completes (i.e., Done at 2017/10/30 29:03:02), and it will only do so when the task is completed with an exit code of 0. It can be a quick way to check if a task is done, by looking at the final line of the standard error log file.

4.3.2 What does the multi-threaded command do

It's very similar to the single-threaded WES solution, except the job will be split evenly based on genomic lengths.

- If you specified "--threads 36," then 36 BED files will be created. Each BED file represents 1/36 of the total base pairs in the human genome (obtained from the .fa.fai file, but only including 1, 2, 3, ..., MT, or chr1, chr2, ..., chrM contigs). They are named 1.bed, 2.bed, ...,

36.bed, and will be created into /ABSOLUTE/PATH/TO/RESULTS/1, /ABSOLUTE/-PATH/TO/RESULTS/2, ..., /ABSOLUTE/PATH/TO/RESULTS/36. You may, of course, specify any number. The default is 12.

- For each mutation callers you specify (with the exception of SomaticSniper), a script will be created into /ABSOLUTE/PATH/TO/RESULTS/1/logs, /ABSOLUTE/PATH/TO/RESULTS/2/logs, etc., with partial BAM input. Again, they will be automatically submitted if you do "--action qsub."
- Because SomaticSniper does not support partial BAM input (one would have to manually split the BAMs in order to parallelize SomaticSniper this way), the above mentioned procedure is not applied to SomaticSniper. Instead, a single-threaded script will be created (and potentially qsub'ed) into /ABSOLUTE/PATH/TO/RESULTS/logs.
  - However, because SomaticSniper is by far the fastest tool there, single-thread is doable even for WGS. Even single-threaded SomaticSniper will likely finish before parallelized Scalpel. When I benchmarked the DREAM Challenge Stage 3 by splitting it into 120 regions, Scalpel took 10 hours and 10 minutes to complete 1/120 of the data. SomaticSniper took a little under 5 hours for the whole thing.
  - After SomaticSniper finishes, the result VCF files will be split into each of the /ABSOLUTE/PATH/TO/RESULTS/1, /ABSOLUTE/PATH/TO/RESULTS/2, etc.
- JointSNVMix2 also does not support partial BAM input. Unlike SomaticSniper, it's slow and takes massive amount of memory. It's not a good idea to run JointSNVMix2 on a WGS data. The only way to do so is to manually split the BAM files and run each separately. We may do so in the future, but JointSNVMix2 is a 5-year old that's no longer being supported, so we probably won't bother.
- Like the single-threaded case, a SomaticSeq run script will also be created for each partition like /ABSOLUTE/PATH/TO/RESULTS/1/SomaticSeq/logs, but will not be submitted until you do so manually.
  - For simplicity, you may wait until all the mutation calling is done, then run a command like

```
find /ABSOLUTE/PATH/TO/RESULTS -name 'somaticseq*.cmd' -exec qsub {} \;
```

5 Use BAMSurgeon to create training set

For your convenience, we have created a couple of wrapper scripts that can generate the run script to create training data using BAMSurgon at somaticseq/utilities/dockered_pipelines/bamSimulator. Descriptions and example commands can be found in the README there.

5.1 Requirements

- Have internet connection, and able to pull and run docker images from docker.io
- Have cluster management system such as Sun Grid Engine, so that the "qsub" command is valid

6 Release Notes

Make sure training and prediction use the same version. Otherwise the prediction is not valid.

6.1 Version 1.0

Version used to generate data in the manuscript and Stage 5 of the ICGC-TCGA DREAM Somatic Mutation Challenge, where SomaticSeq's results were #1 for INDEL and #2 for SNV.

In the original manuscript, VarDict's var2vcf_somatic.pl script was used to generate VarDict VCFs, and subsequently "-filter somatic" was used for SSeq_merged.vcf2tsv.py. Since then (including DREAM Challenge Stage 5), VarDict recommends var2vcf_paired.pl over var2vcf_somatic.pl, and subsequently "-filter paired" was used for SSeq_merged.vcf2tsv.py. The difference in SomaticSeq results, however, is pretty much negligible.

6.2 Version 1.1

Automated the SomaticSeq.Wrapper.sh script for both training and prediction mode. No change to any algorithm.

6.3 Version 1.2

Have implemented the following improvement, mostly for indels:

- SSeq_merged.vcf2tsv.py can now accept pileup files to extract read depth and DP4 (reference forward, reference reverse, alternate forward, and alternate reverse) information (mainly for indels). Previously, that information can only be extracted from SAMtools VCF. Since the SAMtools or HaplotypeCaller generated VCFs hardly contain any indel information, this option improves the indel model. The SomaticSeq.Wrapper.sh script is modified accordingly.
- Extract mapping quality (MQ) from VarDict output if this information cannot be found in SAMtools VCF (also mostly benefits the indel model).
- Indel length now positive for insertions and negative for deletions, instead of using the absolute value previously.

6.4 Version 2.0

- Removed dependencies for SAMtools and HaplotypeCaller during feature extraction. SSeq_merged.vcf2tsv.py extracts those information (plus more) directly from BAM files.
- Allow not only VCF file, but also BED file or a list of chromosome coordinate as input format for SSeq_merged.vcf2tsv.py, i.e., use -mybed or -mypos instead of -myvcf.
- Instead of a separate step to annotate ground truth, that can be done directly by SSeq_merged.vcf2tsv.py by supplying the ground truth VCF via -truth.
- SSeq_merged.vcf2tsv.py can annotate dbSNP and COSMIC information directly if BED file or a list of chromosome coordinates are used as input in lieu of an annotated VCF file.
- Consolidated feature sets, e.g., removed some redunda Fixed a bug: if JointSNVMix2 is not included, the values should be "NaN" instead of 0's. This is to keep consistency with how we handle all other callersnt feature sets coming from different resources.

6.5 Version 2.0.2

- Incorporated LoFreq.
- Used getopt to replace getopts in the SomaticSeq.Wrapper.sh script to allow long options.

6.6 Version 2.1.2

- Properly handle cases when multiple ALT's are calls in the same position. The VCF files can either contain multiple calls in the ALT column (i.e., A,G), or have multiple lines corresponding to the same position (one line for each variant call). Some functions were significantly re-written to allow this.
- Incorporated Scalpel.
- Deprecated HaplotypeCaller and SAMTools dependencies completely as far as feature generation is concerned.
- The Wrapper script removed SnpSift/SnpEff dependencies. Those information can be directly obtained during the SSeq_merged.vcf2tsv.py step. Also removed some additional legacy steps that has become useless since v2 (i.e., score_Somatic.Variants.py). Added a step to check the correctness of the input. The v2.1 and 2.1.1 had some typos in the wrapper script, so only describing v2.1.2 here.

6.7 Version 2.2

- Added MuTect2 support.

6.8 Version 2.2.1

- InDel_3bp now stands for indel counts within 3 bps of the variant site, instead of exactly 3 bps from the variant site as it was previously (likewise for InDel_2bp).
- Collapse MQ0 (mapping quality of 0) reads supporting reference/variant reads into a single metric of MQ0 reads (i.e., tBAM_MQ0 and nBAM_MQ0). From experience, the number of MQ0 reads is at least equally predictive of false positive calls, rather than distinguishing if those MQ0 reads support reference or variant.
- Obtain SOR (Somatic Odds Ratio) from BAM files instead of VarDict's VCF file.
- Fixed a typo in the SomaticSeq.Wrapper.sh script that did not handle inclusion region correctly.

6.9 Version 2.2.2

- Got around an occasional unexplained issue in then ada package were the SOR is sometimes categorized as type, by forcing it to be numeric.
- Defaults PASS score from 0.7 to 0.5, and make them tunable in the SomaticSeq.Wrapper.sh script (--pass-threshold and --lowqual-threshold).

6.10 Version 2.2.3

- Incorporated Strelka2 since it's now GPLv3.
- Added another R script (ada_model_builder_ntChange.R) that uses nucleotide substitution pattern as a feature. Limited experiences have shown us that it improves the accuracy, but it's not heavily tested yet.
- If a COSMIC site is labeled SNP in the COSMIC VCF file, if_cosmic and CNT will be labeled as 0. The COSMIC ID will still appear in the ID column. This will not change any results because both of those features are turned off in the training R script.
- Fixed a bug: if JointSNVMix2 is not included, the values should be "NaN" instead of 0's. This is to keep consistency with how we handle all other callers.

6.11 Version 2.2.4

- Resolved a bug in v2.2.3 where the VCF files of Strelka INDEL and Scalpel clash on GATK CombineVariants, by outputting a temporary VCF file for Strelka INDEL without the sample columns.
- Caller classification: consider if_Scalpel = 1 only if there is a SOMATIC flag in its INFO.

6.12 Version 2.2.5

- Added a dockerfile. Docker repo at https://hub.docker.com/r/lethalfang/somaticseq/.
- Ability to use vcfsort.pl instead of GATK CombineVariants to merge VCF files.

6.13 Version 2.3.0

- Moved some scripts to the utilities directory to clean up the clutter.
- Added the split_Bed_into_equal_regions.py to utilities, which will split a input BED file into multiple BED files of equal size. This is to be used to parallelize large WGS jobs.
- Made compatible with MuTect2 from GATK4.
- Removed long options for the SomaticSeq.Wrapper.sh script because it's more readable this way.
- Added a script to add "GT" field to Strelka's VCF output before merging it with other VCF files. That was what caused GATK CombineVariants errors mentioned in v2.2.4's release notes.
- Added a bunch of scripts at utilities/dockered_pipelines that can be used to submit (requiring Sun Grid Engine or equivalent) dockerized pipeline to a computing cluster.

6.14 Version 2.3.1

- Improve the automated run script generator at utilities/dockered_pipelines.
- No change to SomaticSeq algorithm

6.15 Version 2.3.2

- Added run script generators for dockerized BAMSurgeon pipelines at utilities/dockered_pipelines/bamSurgeon
- Added an error message to r_scripts/ada_model_builder_ntChange.R when TrueVariants_or_False don't have both 0's and 1's. Other than this warning message change, no other change to SomaticSeq algorithm.

6.16 Version 2.4.0

- Restructured the utilities scripts.
- Added the utilities/filter_SomaticSeq_VCF.py script that "demotes" PASS calls to LowQual based on a set of tunable hard filters.
- BamSurgeon scripts invokes modified BamSurgeon script that splits a BAM file without the need to sort by read name. This works if the BAM files have proper read names, i.e., 2 and only 2 identical read names for each paired-end reads.

7 Contact Us

For suggestions, bug reports, or technical support, please post in https://github.com/bioinform/somaticseq/issues. The developers are alerted when issues are created there. Alternatively, you may also email li_tai.fang@roche.com.