SomaticSeq Documentation

May 13, 2016

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, 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 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 (Yes, Python 3. I deserve brownie points from Python developers).
- R, plus the ada package in R.
- BEDTools (if there is an inclusion and/or an exclusion region file)
- Optional: Free GATK (We use it for CombineVariants)
- Optional: dbSNP and COSMIC files in VCF format (if you want to use these featuers as a part of the training).
- At least one of MuTect/Indelocator, VarScan2, JointSNVMix, SomaticSniper, VarDict, MuSE, LoFreq, and/or Scalpel. 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.

2 To use SomaticSeq.Wrapper.sh

The SomaticSeq.Wrapper.sh is a wrapper script that calls a series of programs and procedures. It can be run easily if all the dependencies are met. Some optional parameters are hard-coded as defaults, but can be easily edited by advanced users. In the next section, we also described the workflow in more detail, so you are not 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. You may choose to
    run any combination of callers you like.
# For training, truth file and the correct R script are required.

SomaticSeq.Wrapper.sh \
--mutect             MuTect/variants.snp.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 \
--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 SNV and/or indel if you provide the truth.snp.vcf and/or truth.indel.vcf file(s).

2.2 To predict somatic mutation based on trained classifiers

```
# The *.RData files are trained classifier from the training mode.
SomaticSeq.Wrapper.sh \
--mutect             MuTect/variants.snp.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 \
--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 \
--output-dir         $OUTPUT_DIR
```

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 from different callers, although it does not matter what tools are used to merge VCF files. To make them compatible with GATK, the VCF files are modified. A somatic call is also tagged with the tool names, so the combined VCF retains those information.

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 uses samtools extract sample name information from the BAM files, and then determine which column belongs to the normal, and which column belongs to the tumor.

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

   # If samtools is not in the PATH:
   modify_MuTect.py -infile input.vcf -outfile output.vcf -nbam normal.bam -tbam
       tumor.bam -samtools $PATH/TO/samtools
   ```

   Alternatively, you can supply the normal and tumor sample names, instead of supplying the BAM files:

```
# Modify MuTect's output VCF
# -type snp for MuTect, and -type indel for Indelocator.
modify_MuTect.py -type snp -infile input.vcf -outfile output.vcf -nsm
    NormalSampleName -tsm TumorSampleName
```

2. 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
```

3. 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-liner, 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
```

4. Modify SomaticSniper's output:

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

5. 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.

6. 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
```

7. 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 preserve the tags we have written into each individual VCF file's INFO.

```
# Combine the VCF files for SNV. Any or all of the VCF files may be present.
# -nt 12 means to use 12 threads in parallel
java -jar $PATH/TO/GenomeAnalysisTK.jar -T CombineVariants -R genome.GRCh37.fa
     -nt 12 --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 12 --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 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.

```
# -mutect / -sniper / -varscan / -jsm / -vardict / -muse / -lofreq are optional.
# -truth is also optional, but it is needed to annotate ground truth information,
    and is thus required for model training.
# -dedup will ignore reads that are marked duplicates.
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 -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 -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.

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 --no-save --args Ensemble.sSNV.tsv   < ada_model_builder.R
R --no-save --args Ensemble.sINDEL.tsv < ada_model_builder.R
```

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

Mutation prediction:

```
# Mutation prediction:
R --no-save --args Ensemble.sSNV.tsv.Classifier.RData   Ensemble.sSNV.tsv   Trained
    .sSNV.tsv   < ada_model_predictor.R
R --no-save --args Ensemble.sINDEL.tsv.Classifier.RData Ensemble.sINDEL.tsv Trained
    .sINDEL.tsv < ada_model_predictor.R
```

After mutation prediction, if you feel like it, you may convert Trained.sSNV.tsv and Trained.sINDEL.tsv into VCF files.

```
# 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
# 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.
SSeq_tsv2vcf.py -tsv Trained.sSNV.tsv   -vcf Trained.sSNV.vcf   -pass 0.7 -low 0.1
    -tools CGA VarScan2 JointSNVMix2 SomaticSniper VarDict -all -phred
SSeq_tsv2vcf.py -tsv Trained.sINDEL.tsv -vcf Trained.sINDEL.vcf -pass 0.7 -low 0.1
    -tools CGA VarScan2 VarDict -all -phred
```

4 Release Notes

Make sure training and prediction use the same version.

4.1 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.

4.2 Version 1.1

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

4.3 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.

4.4 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.
- 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 redundant feature sets coming from different resources.

4.5 Version 2.0.2

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

4.6 Version 2.1

- Incorporated Scalpel
- Properly handle calls with multiple ALT calls in the same position. The VCF files can either have more than one call in the ALT column (i.e., A,G), or have multiple lines corresponding to the same position (one line for each call).
- Removed SnpSift/SnpEff dependencies. Those information can be directly obtained during the SSeq_merged.vcf2tsv.py step.

5 To do: planned improvement

- Develop a version for deep sequencing cfDNA/ctDNA.

6 Contact Us

For suggestions, bug reports, or technical support, please email li_tai.fang@bina.roche.com.