SomaticSeq Manual

Li Tai Fang

August 14, 2015

1 SomaticSeq

SomaticSeq is a flexible workflow that uses multiple somatic mutation callers to obtain a combined call set, and then use machine learning to distinguish true mutations from false positives from the call set. The manuscript is in preparation. The source code is deposited at *https://github.com/bioinform/somaticseq/*.

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 files are fed to SomaticSeq.Wrapper.sh, it will either train the call set into a classifier, predict high-confidence somatic mutations from the call set, or do nothing.

2 SomaticSeq.Wrapper.sh Commands

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 ignore, where the ground truth is not known in those regions.

```
# -M/-I/-V/-v/-J/-S/-D/-U are output VCF files from individual
    callers.
# -i is also optional.
SomaticSeq.Wrapper.sh -M MuTect/variants.snp.vcf -I Indelocator/
    variants.indel.vcf -V VarScan2/variants.snp.vcf -v VarScan2/
    variants.indel.vcf -J JointSNVMix2/variants.snp.vcf -S
    SomaticSniper/variants.snp.vcf -D VarDict/variants.vcf -U MuSE/
    variants.snp.vcf -N matched_normal.bam -T tumor.bam -R
    ada_model_builder.R -g human_b37.fasta -c cosmic.b37.v71.vcf -d
     dbSNP.b37.v141.vcf -s $PATH/TO/DIR/snpSift -G $PATH/TO/
    GenomeAnalysisTK.jar -i ignore.bed -Z truth.snp.vcf -z truth.
    indel.vcf -o $OUTPUT_DIR
```

SomaticSeq.Wrapper.sh supports any combination of the somatic mutation callers we have incorporated into the workflow, so -M/-I/-V/-v/-J/-S/-D/-U are all optional parameters. 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 –M MuTect/variants.snp.vcf –I Indelocator/
    variants.indel.vcf –V VarScan2/variants.snp.vcf –v VarScan2/
    variants.indel.vcf –J JointSNVMix2/variants.snp.vcf –S
    SomaticSniper/variants.snp.vcf –D VarDict/variants.vcf –U MuSE/
    variants.snp.vcf –N matched_normal.bam –T tumor.bam –R
    ada_model_predictor.R –C sSNV.Classifier.RData –x sINDEL.
    Classifier.RData –g human_b37.fasta –c cosmic.b37.v71.vcf –d
    dbSNP.b37.v141.vcf –s $PATH/TO/DIR/snpSift –G $PATH/TO/
    GenomeAnalysisTK.jar –o $OUTPUT_DIR
```

3 The step-by-step SomaticSeq Workflow

The SomaticSeq.Wrapper.sh calls a series of programs and procedures. We'll describe the workflow here, so you may modify it for your own needs.

3.1 Combine the call sets

We use GATK CombineVariants to combine the VCF files from different callers. To make them compatible with GATK, the VCFs must be 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. Somatic calls will be attached the tag 'CGA' in the INFO. 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's output VCF
# -type snp for MuTect, and -type indel for Indelocator.
modify_MuTect.py -type snp -infile input.vcf -outfile output
  .vcf -nbam normal.bam -tbam tumor.bam

# If samtools is not in the PATH:
modify_MuTect.py -type snp -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}'

```

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 and StrongSomatic 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:
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
```

   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
```

```

```

8. Use SnpSift to add dbSNP information to the VCF file, since dbSNP information is part of training feature set.

```
java -jar $PATH/TO/SnpSift.jar annotate dbSNP141.vcf
  CombineVariants.snp.vcf > dbSNP.CombineVariants.snp.vcf
java -jar $PATH/TO/SnpSift.jar annotate dbSNP141.vcf
  CombineVariants.indel.vcf > dbSNP.CombineVariants.indel.
  vcf
```

Right now, we do not use COSMIC or functional annotation as a part of training feature, but we do have them in the workflow for "future-proofing." We may decide to use those features in the future when we have better data sets for training.

```
java -jar $PATH/TO/SnpSift.jar annotate cosmic71.vcf dbSNP.
  CombineVariants.vcf  > COSMIC.dbSNP.CombineVariants.vcf
java -jar $PATH/TO/SnpSift.jar GRCh37.75 COSMIC.dbSNP.
  CombineVariants.vcf > EFF.COSMIC.dbSNP.CombineVariants.vcf
```

9. This procedure annotates the caller consensus, use -mincaller 1 to only keep calls where at least one caller has called it somatic. Most calls in the previous files were REJECT or GERMLINE calls. It also does some basic scoring, which are not utilized in SomaticSeq. You will do one for SNV and one for INDEL.

```
# Use -tools to indicate what call sets were combined.
# In this one, 6 tools were used for SNV
score_Somatic.Variants.py -tools CGA VarScan2 JointSNVMix2
  SomaticSniper VarDict MuSE -infile EFF.COSMIC.dbSNP.
  CombineVariants.snp.vcf -mincaller 1 -outfile BINA_somatic
  .snp.vcf

# 3 tools for INDEL
score_Somatic.Variants.py -tools CGA VarScan2 VarDict -
  infile EFF.COSMIC.dbSNP.CombineVariants.indel.vcf -
  mincaller 1 -outfile BINA_somatic.indel.vcf
```