Best Practices for Processing HTS Data Documentation

Release 0.0

Michael Kundsen

Sep 26, 2017

Contents

These workflows are based on the GATK Best Practices. The workflow requires specification of paths to a number of programs and reference datasets which must be downloaded and installed first.

CHAPTER 1

Datasets

- **Reference genome** Analysis requires a number of reference datasets. A ressource bundle with all necessary files for the GATK workflow is provided by the Broad Institute. The link contains bundles for a number of different versions of the human reference genome; in the NEXT bioinformatics network we use the Genomic Data Commons (GDC) version of the GRCh38 reference genome. The reference genome and associated index files is available for download here

CHAPTER 2

Processing of DNA-seq

The workflow below requires specification of paths to a number of datasets and programs:

```
## Datasets
assembly="path to reference genome in .fasta format"
known_snp="path to dbSNP variants in .vcf format"
known_snp_hc="path to high confidence 1000G SNPs in .vcf format"
known_indels="path to Mills and 1000g INDELs in .vcf format"
cosmic="path to .vcf file with known coding and non-coding somatic mutations in the␣
↪COSMIC database"
target_regions="path to bed file with target regions for capture kit"

## Programs
java="path to java 1.8 or newer"
gatk="path to gatk .jar file"
picard="path to picard .jar file"
varscan2="path to varscan2 .jar file"
```

- Alignment

Paired end reads are mapped to a reference genome using BWA MEM:

```
/data/apps/bwa-0.7.12/bwa mem \
-M \
-t28 \
-R"@RG\tID:exome_1\tPL:ILLUMINA\tPU:exome.11\tLB:$id\tSM:$id\tCN:AAU" \
$assembly \
$rawDataDir/$R1 \
$rawDataDir/$R2 \
> aligned_reads.sam
```

- Marking Duplicates

Aligned reads are sorted, converted to BAM, and PCR duplicates are marked:

```
## Sort and convert to bam
$java -jar $picard SortSam \
```

```
INPUT=aligned_reads.sam \
OUTPUT=sorted_reads.bam \
SORT_ORDER=coordinate

## Mark PCR duplicates
$java -jar $picard MarkDuplicates \
INPUT=sorted_reads.bam \
OUTPUT=dedup_reads.bam \
METRICS_FILE=dup_metrics.txt

## Index bam file
$java -jar $picard BuildBamIndex \
INPUT=dedup_reads.bam
\end{lstlisting}
```

- INDEL realignment

Although INDEL realignment is included in the haplotypecaller used in MUTECT2 the reads are realigned in the .bams, so that they look more reasonable when manually validating variants with IGV:

```
## Create a target list of intervals to be realigned
java -jar $gatk \
-T RealignerTargetCreator \
-R $assembly \
-I dedup_reads.bam \
-known $known_indels \
-o realignment_targets.list

## Perform realignment of the target intervals
java -jar $gatk \
-T IndelRealigner \
-R $assembly \
-I dedup_reads.bam \
-targetIntervals realignment_targets.list \
-known $known_indels \
-o realigned_dedup_reads.bam
```

- Base Quality Recalibration

Quality scores in BAM files are recalibrated to adjust for bias in the quality scores which might affect the final variant calls:

```
## Analyze patterns of covariation in the sequence dataset
$java -jar $gatk \
-T BaseRecalibrator \
-R $assembly \
-nct 28 \
-I realigned_dedup_reads.bam \
-knownSites $known_snp \
-knownSites $known_snp_hc \
-knownSites $known_indels \
-o recal_data.table

## Do a second pass to analyze covariation remaining after recalibration
$java -jar $gatk \
-T BaseRecalibrator \
-R $assembly \
-nct 8 \
```

```
-I realigned_dedup_reads.bam \
-knownSites $known_snp \
-knownSites $known_indels \
-BQSR recal_data.table \
-L $regions \
-o post_recal_data.table

## Apply recalibration
$java -jar $gatk \
-T PrintReads \
-R $assembly \
-nct 8 \
-I realigned_dedup_reads.bam \
-BQSR recal_data.table \
-o recal_realigned_dedup_reads.bam
```

- Alignment Metrics

quality metrics for alignment and duplication are calculated using picard tools:

```
$java -jar $picard BedToIntervalList \
I=$target_regions \
O=target_Picard \
SD=$dictionary

$java -jar $picard CollectHsMetrics \
I=recal_realigned_dedup_reads.bam \
O=HSmetrics.txt \
R=$assembly \
TARGET_INTERVALS=target_Picard \
BAIT_INTERVALS=target_Picard
```

- Variant calling for somatic mutations

Somatic variants are called using both Mutect2 and Varscan2, and variants are subsequently merged and filtered. A more detailed description is found in the Mutect2 pitfalls section. Variant calling with Mutect2 can optionally be parallelized by chromosome using the -L parameter for faster runtimes:

```
## Run Mutect2
$java -jar $gatk \
--analysis_type MuTect2 \
--reference_sequence $assembly \
--input_file:normal normal.bam \
--input_file:tumor tumor.bam \
--out $inTumor/somatic_variants.vcf \
--max_alt_alleles_in_normal_count  1000000 \
--max_alt_allele_in_normal_fraction 0.1 \
--cosmic $cosmic \
--dbsnp $known_snp \
-nct 28
```

Variant calling with varscan2 requires an mpileup file which can be built with samtools using the aligned BAM files for tumor and normal samples:

```
## Build mpileup with samtools
samtools mpileup \
-f $assembly \
-q 1 \
```

```
-B normal.bam \
tumor.bam > normal-tumor.mpileup
```

Variants may then be called with varscan2 and high confidence SNPs/INDELs can be extracted usign the processSomatic command:

```
## Run varscan2 somatic
$java -jar $varscan2 \
somatic \
normal-tumor.mpileup \
tumor_variants.varscan2 \
--mpileup 1 \
--min-var-freq 0.02 \
--output-vcf

## Process SNPs
$java -jar $varscan2 \
processSomatic \
tumor_variants.varscan2.snp.vcf

## Process INDELs
$java -jar $varscan2 \
processSomatic \
tumor_variants.varscan2.indel.vcf
```

- Variant filtration

Final set of somatic SNPs / INDELS are found by combining and filtering outputs from Mutect2 and varscan2 as described in the Mutect2 pitfalls section. Briefly, for a variant to pass filtering the following must be fulfilled:

```
1) PASS in Mutect2 or called by MuTect2 + PASS in varscan2 HC
2) Tumor AF > 4 * Normal AF
3) QSS / AD > 25
```

CHAPTER 3

Processing of RNA-seq

- SDU

The SDU workflow for processing RNA-seq data is given as:

```
#!/bin/sh

#### PATHS TO REQUIRED SOFTWARE, INPUT & OUTPUT FOLDERS
DAT="/path/to/data" ### PATH TO FOLDER CONTAINING GZ-COMPRESSED FASTQ FILES
OUT="/path/to/output" ### PATH TO FOLDER DEPOSITING THE RESULTS
REFT="/path/to/reference" ### PATH TO FOLDER TRANSCRIPTOME REFERENCE (gtf/gff file)
REFG="/path/to/reference" ### PATH TO FOLDER GENOME REFERENCE (fa file and bowtie␣
↪index files)
SAMTOOLS="/path/to/samtools" ### PATH TO SAMTOOLS
TOPHAT="/path/to/tophat2" ### PATH TO TOPHAT2
BOWTIE="/path/to/bowtie" ### PATH TO BOWTIE
BBMAP="/path/to/BBMAP" ### PATH TO BBMAP
PYTHON="/path/to/python"
HTSEQ="/path/to/HTSeq"

export PATH=$SAMTOOLS:$BOWTIE:$TOPHAT:$HTSEQ:$PATH
echo $PATH

### UNPACKING gz-compressed fastq files
cd $DAT
for i in *_R1.fastq.gz;
do
newfile=$(basename $i _R1.fastq.gz)
gunzip /$DAT/${newfile}_R1.fastq.gz
gunzip /$DAT/${newfile}_R2.fastq.gz
done

### Quality cleaning (adaptor removal, trimming of low quality bases and reads)
for i in *_R1.fastq;
do
newfile=$(basename $i _R1.fastq)
```

```
$BBMAP/bbduk.sh -Xmx20g \
                in1=/$DAT/${newfile}_R1.fastq \
                in2=/$DAT/${newfile}_R2.fastq \
                out1=/$DAT/${newfile}_clean_R1.fastq \
                out2=/$DAT/${newfile}_clean_R2.fastq \
                ref=$BBMAP/resources/adapters.fa \
                ktrim=r ktrim=l k=23 mink=11 hdist=1 tpe tbo \
                qtrim="rl" trimq=10 maq=10 minlen=25
done

#### Bowtie2-Tophat2 alignment
for i in *_clean_R1.fastq;
do
newfile=$(basename $i _clean_R1.fastq)
$TOPHAT -p 1 -G $REFT \
                --output-dir $OUT/${newfile} \
                $REFG $DAT/${newfile}_clean_R1.fastq \
                $DAT/${newfile}_clean_R2.fastq
$SAMTOOLS/samtools index $OUT/${newfile}/accepted_hits.bam
mv $OUT/${newfile}/accepted_hits.bam $OUT/${newfile}/${newfile}_accepted_hits.bam
mv $OUT/${newfile}/accepted_hits.bam.bai $OUT/${newfile}/${newfile}_accepted_hits.bam.
↪bai

### Count Matrix construction by HTSeq
$python -m HTSeq.scripts.count \
                --format bam \
                --mode union \
                --stranded no \
                --minaqual 1 \
                --type gene \
                --idattr gene_id \
                $OUT/${newfile}/${newfile}_accepted_hits.bam $REFT \
                > $OUT/${newfile}_gene_read_counts_table.tsv

done
```

- AAUH

  The spliced alignment of RNA-seq performed with tophat in the above script can altertively be done using a 2-pass alignment with STAR. This is the suggested method in the GATK best practices

A script for doing this is shown below:

```
## Create temp dir
mkdir /scratch/$PBS_JOBID
TMPDIR=/scratch/$PBS_JOBID
cd $TMPDIR

## Path to paired fastq files
rawDataDir="Path to directory with QC fastq files"
R1=$(ls $rawDataDir | grep $id | grep R1)
R2=$(ls $rawDataDir | grep $id | grep R2)

## Move fastq files to scratch
cp $rawDataDir/$R1 $rawDataDir/$R2 $TMPDIR

## Programs
STAR="path to star"
```

```
## Reference data
assembly="Path to reference genome fasta"
genomeDir="Path to STAR indexed reference genome"

#########################
#### Align using STAR ####
#########################

### Do 1st pass
mkdir $TMPDIR/1pass
cd $TMPDIR/1pass

$STAR \
--genomeDir $genomeDir \
--readFilesIn ../$R1 ../$R2 \
--readFilesCommand zcat \
--runThreadN 8

### Create new index using splice junction information from 1st pass
mkdir $TMPDIR/b37_2pass

$STAR \
--runMode genomeGenerate \
--genomeDir $TMPDIR/b37_2pass \
--genomeFastaFiles $assembly \
--sjdbFileChrStartEnd $TMPDIR/1pass/SJ.out.tab \
--sjdbOverhang 75 \
--genomeSAsparseD 2 \
--runThreadN 8 \
--limitGenomeGenerateRAM 20000000000

### Do 2nd pass
mkdir $TMPDIR/2pass
cd $TMPDIR/2pass
$STAR \
--genomeDir $TMPDIR/b37_2pass \
--readFilesIn ../$R1 ../$R2 \
--readFilesCommand zcat \
--runThreadN 8 \
--outSAMstrandField intronMotif \
--outSAMtype BAM SortedByCoordinate \
--outFileNamePrefix ${id}_STAR_
#--outSaMmapqUnique 60 \

## Return output
cp * $outDir

## Clean up scratch
cd /scratch
rm -fr $PBS_JOBID
```

Transcipt level expression can then be inferred using Cufflinks. This is done using the script below:

```
## Paths
mkdir /scratch/$PBS_JOBID
TMPDIR=/scratch/$PBS_JOBID
bamFile="path to STAR aligned BAM file"
cufflinks="path to cufflinks"
```

```
gff="Path to gff file"
outDir="Path for output files"

cd $TMPDIR
cp $bamFile $TMPDIR

# Construct the mask file
grep rRNA $gff > mask.gff3
grep tRNA $gff >> mask.gff3

# Run cufflinks
echo Running cufflinks ...
$cufflinks \
--GTF-guide $gff \
--mask-file mask.gff3 \
--library-type fr-unstranded \
--num-threads 8 \
--output-dir cufflinks \
--quiet \
$bamFile

echo moving files to home dir
cp -fr $TMPDIR/cufflinks/* $outDir
```

CHAPTER 4

Structural Variants

- SDU

The following script developed at SDU is used to infer breakpoints from DNA-seq data with Breakdancer:

```
#!/bin/sh
### Specify paths to requiered software, input and output folders
BWA="/path/to/bwa"
picard="/path/to/picard-tools"
samtools="/path/to/samtools"
DAT="/path/to/fastqfiles/"
OUTPUT="/path/to/outputfolder"
REF="/path/to/reference"
BBMAP="/path/to/bbmap"
BREAKDANCER="/path/to/breakdancer"

###### Unpacking compressed fastq files
cd $DAT
for i in *_R1.fastq.gz;
do
newfile=$(basename $i _R1.fastq.gz)
gunzip $DAT/${newfile}_R1.fastq.gz
gunzip $DAT/${newfile}_R2.fastq.gz
done

###### Quality filtering
for i in *_R1.fastq;
do
newfile=$(basename $i _R1.fastq)
$BBMAP/bbduk.sh -Xmx20g \
                in1=/$DAT/${newfile}_R1.fastq \
                in2=/$DAT/${newfile}_R2.fastq \
                out1=/$DAT/${newfile}_clean_R1.fastq \
                out2=/$DAT/${newfile}_clean_R2.fastq \
                ref=$BBMAP/resources/adapters.fa \
                ktrim=r ktrim=l k=23 mink=11 hdist=1 \
```

```
                tpe tbo qtrim="rl" trimq=10 maq=10 minlen=25
done

###### BWA alignment
for i in *_clean_R1.fastq;
do
newfile=$(basename $i _clean_R1.fastq)
$BWA mem -t 4 -M -R '@RG\tID:${newfile}.lib.run\tLB:${newfile}.lib\tPL:ILLUMINA\tSM:$
↪{newfile}' \
$REF/hg19.fa $DAT/${newfile}_clean_R1.fastq $DAT/${newfile}_clean_R2.fastq > $OUTPUT/$
↪{newfile}.sam

### Samtools processing of aligned reads
$samtools view -bS -@ 4 $OUTPUT/${newfile}.sam > $OUTPUT/${newfile}.bam

rm $OUTPUT/${newfile}.sam

$samtools sort $OUTPUT/${newfile}.bam -o $OUTPUT/${newfile}_sorted.bam

$samtools index $OUTPUT/${newfile}_sorted.bam $OUTPUT/${newfile}_sorted.bai

### PCR duplicate removal
java -jar $picard/MarkDuplicates.jar \
                INPUT=$OUTPUT/${newfile}_sorted.bam \
                OUTPUT=$OUTOUT/${newfile}_sorted_nodup.bam \
                METRICS_FILE=${newfile}_dup.metrics \
                REMOVE_DUPLICATES=TRUE \
                VALIDATION_STRINGENCY=LENIENT
$samtools index $OUTPUT/${newfile}_sorted_nodup.bam $OUTPUT/${newfile}_sorted_nodup.
↪bai
rm $OUTPUT/${newfile}_sorted.bam
rm $OUTPUT/${newfile}_sorted.bai

done

### Run breakdancer
cpanm --local-lib=~/perl5 local::lib && eval $(perl -I ~/perl5/lib/perl5/ -
↪Mlocal::lib)
cd $OUTPUT
for i in *_sorted_nodup.bam;
do
newfile=$(basename $i _sorted_nodup.bam)
perl $BREAKDANCER/bam2cfg.pl $OUTPUT/${newfile}_sorted_nodup.bam  > $OUTPUT/${newfile}
↪.cfg
perl $BREAKDANCER/BreakDancerMax.pl $OUTPUT/${newfile}.cfg -d -g -h -y > $OUTPUT/$
↪{newfile}.ctx

done
```

- AAUH

Fusion genes may be inferred from RNA-seq data using FusionMap. Running Fusionmap on a Linux machine requires installing Mono; instructions can be found on the FusionMap website.

Fusionmap Requires a control file, with the following specifications:

```
MonoPath="path to mono"
PairedEnd=True  //Automatically pair two fastq files as one sample to run fusion␣
↪analysis
```

```
RnaMode=True    //Detect fusion results
ThreadNumber=8  //Possible values: 1-100. Default value=1
FileFormat=FASTQ //Possible values: FASTQ, QSEQ, FASTA, BAM. Default value=FASTQ
CompressionMethod=Gzip   //Gzip formatted input files
Gzip=True        //Gzip
OutputFusionReads=True //Out put Fusion reads as BAM files for genome browser.
↪Default value=False

<Output>
TempPath=temp
OutputPath=output
OutputName=OUT.file
```

FusionMAP can then be run using the script below:

```
## Set paths
mkdir /scratch/$PBS_JOBID
TMPDIR=/scratch/$PBS_JOBID
cd $TMPDIR
rawDataDir="Path to QC fastQ files"
outDir="Path for output"
controlDir="path to control file"
mono="path to Mono"
fusionmap="path to fusionMap"

## Paired fastq files
R1=$(ls $rawDataDir | grep fastq | grep R1)
R2=$(ls $rawDataDir | grep fastq | grep R2)

## Move fastq files to scratch on node
cp $rawDataDir/$R1 $rawDataDir/$R2 $TMPDIR

cd $TMPDIR

## Prepare control file
sed s/R1.file/$R1/g $controlDir/fusionmap.control.txt |\
        sed s/R2.file/$R2/g | sed s/OUT.file/$id/g > fusion.control

## Run FusionMap
$mono $fusionmap \
        --semap /data/apps/software/FusionMap_2015-03-31 \
        Human.B37.3 RefGene \
        fusion.control

mv -fr $TMPDIR/output $outDir
rm -fr $TMPDIR
```

CHAPTER 5

CLC Germline workflow

- **fastq files are imported to clc.**
  - Reads that did not pass a quality filter are ignored.
- **Trimming sequence ends.**
  - **Quality trimming:**
    - Phred score > 20
    - trim ambiguous nucleotides
  - **Sequence filtering:**
    - Remove 1 3' terminal nucleotide
    - number of residues of reads must be in that range: [30, 500]; short and long reads are discarded.
- **Map reads to reference (hg19). For algorithm see: http://www.clcbio.com/files/whitepapers/whitepaper-on-CLC-read-ma**
  - **Mapping parameters:**
    - match score 1
    - mismatch cost 2
    - linear gap cost
    - insertion cost 3
    - deletion cost 3
    - length fraction 0.95
    - similarity fraction 0.95
    - global alignment
    - ignore non-specific matches.
- **Local Realignment.**

  - Realign unaligned ends
  - perform local realignment 2 times.

- **Variant Detection. See:** http://www.clcbio.com/files/whitepapers/whitepaper-probabilistic-variant-caller-1.pdf

  - Minimum coverage:10
  - min count: 3
  - min frequency: 25.
  - Restrict calling to target region: as given by SureSelect.

CHAPTER 6

MuTect2 Pitfalls

The standard parameters in MuTect2 (GATK v3.7) are very strict. For example, a somatic variant (no matter its frequency in the tumor sample) is discarded if the variant is seen in more than one read in the normal sample. This is particularly problematic for panel sequencing with read depths >1000X in targeted regions.

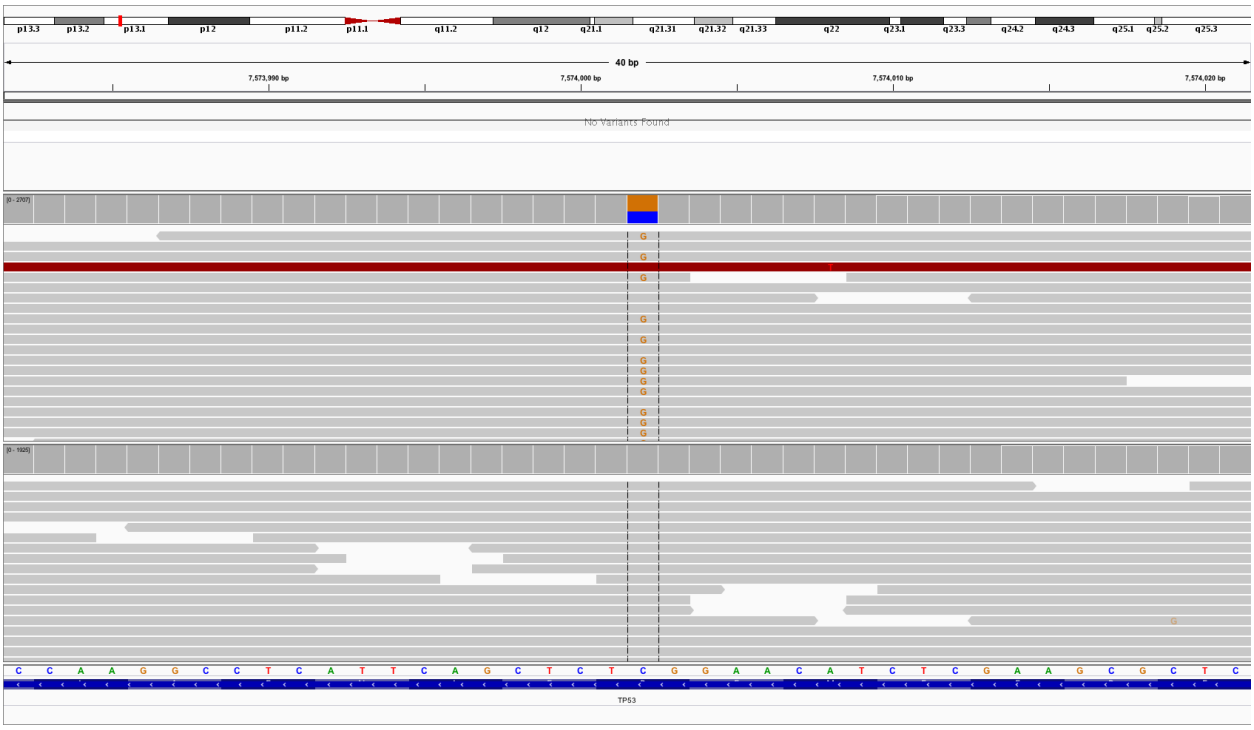
We propose solving this issue by raising the values of two MuTect2 parameters as indicated in the table below. By setting *max_alt_alleles_in_normal_count* to a very high number (there is no option to completely disable this filter), we never discard variants based on the absolute count of ALT alleles in the normal sample. Furthermore, we allow up to 10% of the normal reads to contain the ALT allele.

Parameter	Default	Proposed
max_alt_alleles_in_normal_count	1	10000000
max_alt_allele_in_normal_fraction	0.03	0.10

These more relaxed settings inevitable lead to an increased number of false positives. We filter those using custom filters (described below).

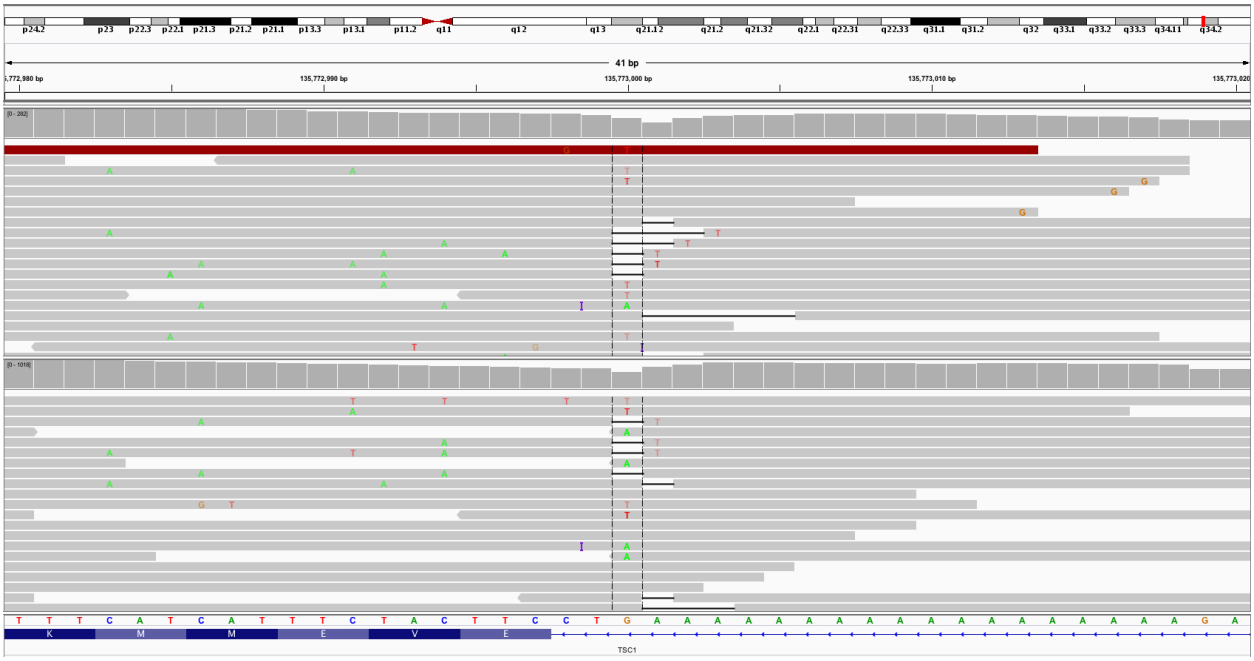
Motivation

The IGV screenshot below shows an example of a variant in TP53 which is not called by MuTect2 with default parameters. The allele frequency in the tumor sample is 58% (1528/2634). However, the variant is filtered out by MuTect2, because it fails both filters above. The allele frequency in the normal sample is 1.1% (21/1920).



Downstream Filters

MuTect2 with standard makes many false positive calls in noisy regions, and this only gets worse with the relaxed paramter settings. For example, the TSC1 variant in the screenshot below is not filtered by MuTect2, although it is clearly noise.



Another typical cause of false positives is similar allele frequencies in tumor and normal. MuTect2 will call variants in the tumor sample as somatic, as long as their frequencies in the normal sample are not above the 10% cut-off. In some

cases, we therefore end up calling variants with higher allele frequencies in the normal than in the tumor as somatic.

To deal with the issue of false positives, we propose the following two metrics calculated from info in the MuTect2 VCF files:

$$S_{AF} = \begin{cases} 1 - \frac{AF_{normal}}{AF_{tumor}} & \text{if} \quad AF_{tumor} > AF_{normal} \\ 0 & \text{otherwise} \end{cases}$$

$$S_{QSS} = \begin{cases} \frac{QSS_{tumor}}{AD_{tumor}} & \text{if} \quad AD_{tumor} > 0 \\ 0 & \text{otherwise} \end{cases}$$

We have found that filtering samples *not* meeting these three criteria:

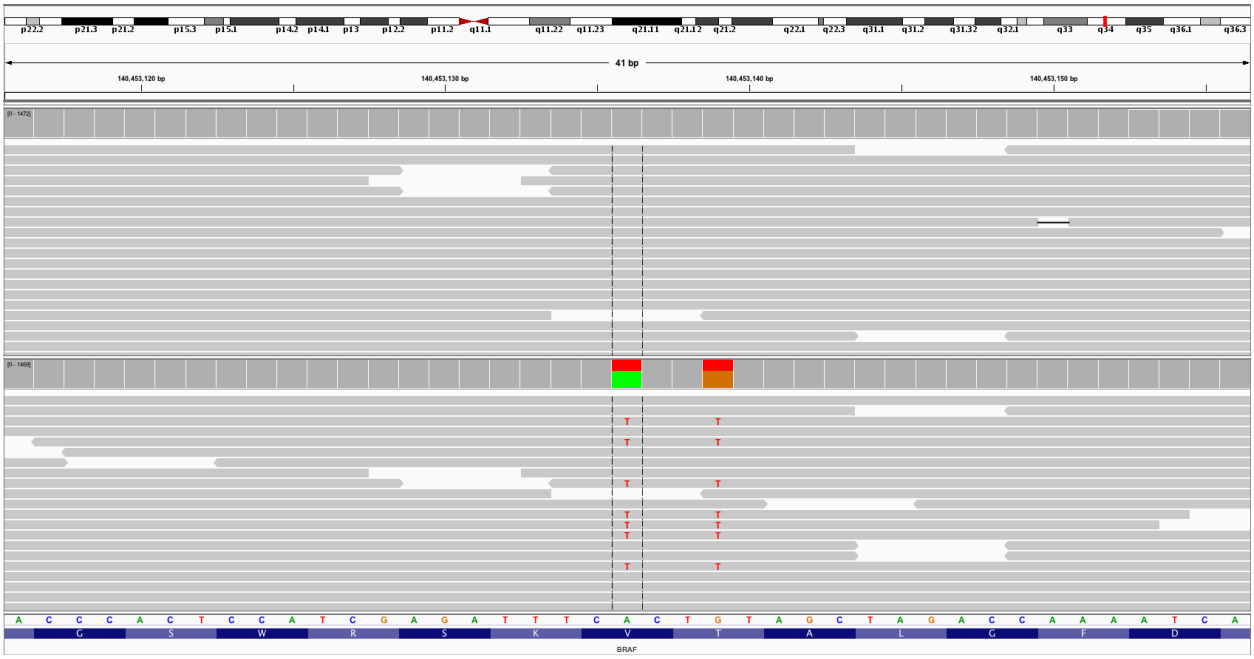
$$AF_{tumor} > 0.02, \quad S_{QSS} > 20, \quad \text{and} \quad S_{AF} > 0.75$$

effectively eliminates all false positives and never removes genuine variants.

The rationale behind these values is as follows: The QSS score is the average base quality of variant bases. If this is below 20, the region is very noisy, and the variant is not likely to be genuine. Furthermore, note that $S_{AF} > 0.75$ is equivalent to $AF_{tumor} > 4 \cdot AF_{normal}$, which means that a variant with a "high" frequency in the normal sample is not called as somatic, unless its frequency in the tumor sample is "much" higher.

Adding Second Variant Caller

Despite the optimization of MuTect2 described above, important variants may still be missed. The screenshot below shows a clear hotspot variant (BRAF V600E) filtered out by MuTect2. This particular variant has been independently validated using a complimentary method.



In fact, MuTect2 not only filters out V600E but also a variant in the neighboring codon. The explanation for both variants being filtered is indicated as *clustered_events* and *multi_event_alt_allele_in_normal* in the VCF file.

Simply disabling these filters leads to an explosion in false positive calls. Instead, we have found that using an additional variant caller besides MuTect2 may rescue erroneously filtered variants. More precisely, we run VarScan2

and check if any *high confidence* variants were also called (but subsequently filtered) by MuTect2. If so, we keep these in a separate file for manual inspection. In the particular example above, both BRAF variants are rescued with no extra false positives added.