|  |  |  |
| --- | --- | --- |
| **{ Title }** | | |
| **Responsible Department: CLS-NGS** | **Effective Date: { Publication Date }** | |
| **Policy Basis for Procedure**  This SOP describes the Alignment, Tumor Only and Somatic Mutational Profiling workflows used for NGS analysis in the UTSW Genomics and Molecular Pathology Lab. First, genomic data is aligned to the human genome, this alignment is optimized to improve quality. Quality metrics on the sequence and alignment are calculated. All single nucleotide variants (SNV), insertion and deletion (Indel) variants are identified in genome/exome/target panel sequencing data. When a normal control from the same patient is provided in the form of genomic DNA from blood or saliva, somatic single nucleotide variants (SNV), insertion and deletion (Indel) variants are identified to provide better sensitivity. | | |
| **Applicability**  This applies to the bioinformatics personnel of the NGS Lab. | | |
| **Description of Standard Procedure**  **DATA REQUIREMENT**  The pipeline requires as input FASTQ files generated by genome/exome/target panel sequencing. The pipeline also requires the following reference data:  (1) the FASTA sequence of a reference genome, currently Human GRCh38 <https://github.com/lh3/bwa/tree/master/bwakit> ;  (2) the target enrichment regions  (3) a set of reference single nucleotide polymorphisms (SNPs) from the NCBI SNP database, currently dbSNP\_BUILD\_ID=150; <https://www.ncbi.nlm.nih.gov/projects/SNP/snp_summary.cgi?view+summary=view+summary&build_id=150>  (4) a set of somatic mutations from COSMIC database  <http://cancer.sanger.ac.uk/cosmic/download> and  (5) a set of reference indels from the 1000 genomes project  <https://github.com/snewhouse/ngs_nextflow/wiki/GATK-Bundle>.  The Reference genome was downloaded and indexed using the commands:   1. bwa.kit/run-gen-ref hs38DH 2. bwa.kit/bwa index hs38DH.fa   **QUALITY CONTROL**  Assure that all quality metrics pass threshold.  Key quality metric thresholds for evaluating the coverage level achieved for an individual tumor sample will be a 1) map rate greater than 90%, 2) properly paired reads greater than 80%, 3) target reads greater than 40%, 4) minimum average read depth > 500X.  For tumor/normal pairs, samples are required to pass correlation analysis.  Sample will be automatically flagged as failed if map rate falls below 90% and deduplication percent greater than 100x falls below 95%.  An exception report will be created for any metrics that does not reach threshold. Corrective actions, further processing or re-processing will be determined by the medical advisor. Corrective action could include but are not limited to analysis error log evaluations, re-sequencing of sample or examination of laboratory quality metrics.  **PROCEDURE**  **Alignment Procedure**        Figure 1: Alignment Overview  1. Trim Sequences  Trim the ends of sequences with remaining adapter or quality scores < 25. Remove any sequence less than 35bp after trimming. Gather stats on trimmed sequences.   |  |  | | --- | --- | | executable | trimgalore.sh parse\_trimreport.pl trimgalore/0.4.1 cutadapt/1.9.1 | | Input | Read1 (${R1}), Read2 (${R2}) gzip fastq files | | Output | trimmed and gzipped fastq files and report of number of reads remaining post trimming | | command | [] trim\_galore --paired -q 25 --illumina --gzip --length 35 ${R1} ${R2}  [] mv ${r1base}\_val\_1.fq.gz ${pair\_id}.trim.R1.fastq.gz  [] mv ${r2base}\_val\_2.fq.gz ${pair\_id}.trim.R2.fastq.gz  [] perl parse\_trimreport.pl ${pair\_id}.trimreport.txt \*trimming\_report.txt |   2. Read Alignment to Human Reference Genome  Align sequencing reads to human reference genome considering ALT contigs, add umi sequence to header, sort alignments, and add mate information.   |  |  | | --- | --- | | executable | dnaseqalign.sh python/2.7.x-anaconda bwakit/0.7.15 samtools/gcc/1.8 picard/2.10.3 bwa/intel/0.7.17 | | Input | Read 1 ${fq1}and Read 2 ${fq2} gzipped fastq files, reference fasta, reference ALT | | Output | umi tagged and sorted bam files | | command | [] bwa mem -M -t $NPROC -R @RG\tID:${read\_group }\tLB:tx\tPL:illumina\tPU:barcode\tSM:${read\_group}" ${index\_path}/genome.fa ${fq1} ${fq2} > out.sam  [] k8 ${testexe}/bwa-postalt.js -p tmphla ${index\_path}/genome.fa.alt out.sam | python ${baseDir}/add\_umi\_sam.py -s - -o output.unsort.bam  [] samtools sort -n --threads $NPROC -o output.dups.bam output.unsort.bam  [] java -Djava.io.tmpdir=./ -Xmx4g -jar $PICARD/picard.jar FixMateInformation ASSUME\_SORTED=TRUE SORT\_ORDER=coordinate ADD\_MATE\_CIGAR=TRUE I=output.dups.bam O=${pair\_id}.bam  [] samtools index ${pair\_id}.bam |   3. Consensus Alignments  Group reads based on umi, generate consensus reads and realign consensus reads to human reference genome.   |  |  | | --- | --- | | executable | markdups.sh picard/2.10.3 fgbio bwakit/0.7.15 bwa/intel/0.7.17 samtools/gcc/1.8 | | input | Aligned bam, reference fasta, genome file | | output | Consensus bam | | command | [] samtools index -@ $NPROC ${sbam}  [] fgbio –tmp-dir ./ GroupReadsByUmi -s identity -i ${sbam} -o group.bam –family-size-histogram ${pair\_id}.umihist.txt –e 0 –m 0  [] fgbio –tmp-dir ./ CallMolecularConsensusReads -i ${pair\_id}.group.bam -p consensus -M 1 -o ${pair\_id}.consensus.bam -S ':none:'  [] samtools index -@ $NPROC ${pair\_id}.consensus.bam  [] samtools fastq -1 ${pair\_id}.consensus.R1.fastq -2 ${pair\_id}.consensus.R2.fastq ${pair\_id}.consensus.bam  [] gzip ${pair\_id}.consensus.R1.fastq  [] gzip ${pair\_id}.consensus.R2.fastq  [] bwa mem -M -C -t 2 -R "@RG\tID:${pair\_id}\tLB:tx\tPL:illumina\tPU:barcode\tSM:${pair\_id}" ${index\_path}/genome.fa ${pair\_id}.consensus.R1.fastq.gz ${pair\_id}.consensus.R2.fastq.gz > out.sam  [] k8 ${testexe}/bwa-postalt.js -p tmphla ${index\_path}/genome.fa.alt out.sam | samtools view -1 - > ${pair\_id}.consensus.bam  [] samtools sort --threads $NPROC -o ${pair\_id}.dedup.bam ${pair\_id}.consensus.bam  [] samtools sort --threads $NPROC -o ${pair\_id}.group.bam group.bam  [] samtools index -@ $NPROC ${pair\_id}.group.bam  [] mv ${pair\_id}.dedup.bam ${pair\_id}.consensus.bam  [] mv ${pair\_id}.dedup.bam.bai ${pair\_id}.consensus.bam.bai |   4. Calculate Metrics of Quality of Alignments  Estimate consensus coverage of bases by exon.   |  |  | | --- | --- | | executable | bamqc.sh samtools/gcc/1.8 fastqc/0.11.5 bedtools/2.26.0 picard/2.10.3 | | input | Consensus bam | | output | alignment stats values including mapping rate, on-target rate, coverage statistics, insert size, library complexity, quality and alignment summary metrics | | command | [] samtools flagstat ${sbam} > ${pair\_id}.flagstat.txt  [] fastqc -f bam ${sbam}  [] samtools view -@ $NPROC -b -L ${bed} -o ${pair\_id}.ontarget.bam ${sbam}  [] samtools index -@ $NPROC ${pair\_id}.ontarget.bam  [] samtools flagstat ${pair\_id}.ontarget.bam > ${pair\_id}.ontarget.flagstat.txt  [] java -Xmx64g -Djava.io.tmpdir=${tmpdir} -jar $PICARD/picard.jar CollectAlignmentSummaryMetrics R=${index\_path}/genome.fa I=${pair\_id}.ontarget.bam OUTPUT=${pair\_id}.alignmentsummarymetrics.txt TMP\_DIR=${tmpdir}  [] java -Xmx64g -Djava.io.tmpdir=${tmpdir} -XX:ParallelGCThreads=$NPROC -jar $PICARD/picard.jar EstimateLibraryComplexity I=${sbam} OUTPUT=${pair\_id}.libcomplex.txt TMP\_DIR=${tmpdir}  [] samtools view -@ $NPROC -b -q 1 ${sbam} | bedtools coverage -sorted -hist -g ${index\_path}/genomefile.txt -b stdin -a ${bed} > ${pair\_id}.mapqualcov.txt  [] samtools view -@ $NPROC ${sbam} | awk '{sum+=$5} END { print "Mean MAPQ =",sum/NR}' > ${pair\_id}.meanmap.txt  [] java -Xmx64g -Djava.io.tmpdir=${tmpdir} -jar $PICARD/picard.jar CollectInsertSizeMetrics INPUT=${sbam} HISTOGRAM\_FILE=${pair\_id}.hist.ps REFERENCE\_SEQUENCE=${index\_path}/genome.fa OUTPUT=${pair\_id}.hist.txt TMP\_DIR=${tmpdir}  [] bedtools coverage -sorted -g ${index\_path}/genomefile.txt -a ${bed} -b ${sbam} -hist > ${pair\_id}.covhist.txt  [] grep ^all ${pair\_id}.covhist.txt > ${pair\_id}.genomecov.txt  [] perl calculate\_depthcov.pl ${pair\_id}.covhist.txt |   5. Identify Viral Reads  Extract non-human reads from BAM file, re-align reads to viral genomes, gather alignment stats.   |  |  | | --- | --- | | executable | viralalign.sh bwa/intel/0.7.17 picard/2.10.3 samtools/1.6 | | input | BAM file | | output | Viral BAM file, Flagstat and idxstats output files | | command | [] samtools view -@ 8 -b -u -F 2 ${bam} |samtools sort -n - >unmapped.bam  [] java -Djava.io.tmpdir=./ -Xmx4g -jar $PICARD/picard.jar SamToFastq I=unmapped.bam FASTQ=unmapped.R1.fastq SECOND\_END\_FASTQ=unmapped.R2.fastq UNPAIRED\_FASTQ=unmapped.unpaired.fastq  [] bwa mem -M -t 4 -R "@RG\tID:${pairid}\tLB:tx\tPL:illumina\tPU:barcode\tSM:${pairid}" ${reffa} unmapped.R1.fastq unmapped.R2.fastq > out.sam  [] samtools view -h -F 256 -b out.sam -o out.bam  [] samtools sort out.bam -o ${pairid}.viral.bam  [] samtools index ${pairid}.viral.bam  [] samtools idxstats ${pairid}.viral.bam >${pairid}.viral.idxstats.txt  [] samtools flagstat ${pairid}.viral.bam >${pairid}.viral.flagstat.txt  [] perl combine\_idxstats.pl \*.viral.idxstats.txt –p ${pairid} |   6. Calling Copy Number Variations  Identify copy number variations   |  |  | | --- | --- | | executable | cnvkit.sh cnvkit/0.9.5 bedtools/2.26.0 samtools/gcc/1.8 bcftools/gcc/1.8 java/oracle/jdk1.8.0\_171 snpeff/4.3q filter\_cnvkit.pl | | input | sorted bam | | output | Copy number ratio files(.cnr), segmented log2 ratio estimates(.cns),cnvcalls, Answer and cbioportal formatted cnvcalls. Scatter plot of all cnvcalls | | command | [] cnvkit.py coverage ${sbam} ${targets}targets.bed -o ${pair\_id}.targetcoverage.cnn  [] cnvkit.py coverage ${sbam} ${targets}antitargets.bed -o ${pair\_id}.antitargetcoverage.cnn  [] cnvkit.py fix ${pair\_id}.targetcoverage.cnn ${pair\_id}.antitargetcoverage.cnn ${normals} -o ${pair\_id}.cnr  []if [[ $panelsize -gt 4000000 ]] cnvkit.py segment ${pair\_id}.cnr -o ${pair\_id}.cns  else cnvkit.py segment -m haar ${pair\_id}.cnr -o ${pair\_id}.cns  []bcftools mpileup -A -d 1000000 -C50 -Ou --gvcf 0 -f ${reffa} -a INFO/AD,INFO/ADF,INFO/ADR,FORMAT/DP,FORMAT/SP,FORMAT/AD,FORMAT/ADF,FORMAT/ADR -T ${index\_path}/IDT\_snps.hg38.bed ${sbam} | bcftools call -m --gvcf 0 -Ov | bcftools convert --gvcf2vcf -f ${reffa} -Ov -o common\_variants.vcf  [] formatVcfCNV.pl cnvkit\_common common\_variants.vcf  [] echo -e "CHROM\tPOS\tAO\tRO\tDP\tMAF" > ${pair\_id}.ballelefreq.txt  [] java -jar $SNPEFF\_HOME/SnpSift.jar extractFields cnvkit\_common.vcf CHROM POS GEN[0].AO GEN[0].RO GEN[0].DP |grep -v CHROM | awk '{print $1"\t"$2"\t"$3"\t"$4"\t"$5"\t"$3/$5}' >> ${pair\_id}.ballelefreq.txt  [] cnvkit.py call --filter cn ${pair\_id}.cns -v cnvkit\_common.vcf -o ${pair\_id}.call.cns  [] cnvkit.py scatter ${pair\_id}.cnr -s ${pair\_id}.call.cns -t --segment-color "blue" -o ${pair\_id}.cnv.scatter.pdf -v cnvkit\_common.vcf  [] cut -f 1,2,3 ${pair\_id}.call.cns | grep -v chrom | bedtools intersect -wao -b ${index\_path}/cytoBand.txt -a stdin |cut -f 1,2,3,7 > ${pair\_id}.cytoband.bed  [] perl filter\_cnvkit.pl -s ${pair\_id}.call.cns |   7. SV Calling itdseek  Identify structural variants using itdseek   |  |  | | --- | --- | | executable | svcalling.sh htslib/gcc/1.8 samtools/gcc/1.8 bcftools/gcc/1.8 bedtools/2.26.0 snpeff/4.3q vcftools/0.1.14 filter\_itdseeker.pl | | input | BAM file | | output | Tandem duplication vcf file | | command | [] samtools view -@ $NPROC -L ${bed} ${sbam} | itdseek.pl --refseq ${reffa} --samtools ${stexe} --bam ${sbam} | vcf-sort | bedtools intersect -header -b ${bed} -a stdin | java -Xmx30g -jar $SNPEFF\_HOME/SnpSift.jar filter "( LEN < 10000 )" | bgzip > ${pair\_id}.itdseek.vcf.gz  [] tabix ${pair\_id}.itdseek.vcf.gz  [] bcftools norm --fasta-ref $reffa -m - -Ov ${pair\_id}.itdseek.vcf.gz | java -Xmx30g -jar $SNPEFF\_HOME/snpEff.jar -no-intergenic -lof -c $SNPEFF\_HOME/snpEff.config ${snpeffgeno} - |bgzip > ${pair\_id}.itdseek\_tandemdup.vcf.gz  [] perl filter\_itdseeker.pl -t ${pair\_id} -d ${pair\_id}.itdseek\_tandemdup.vcf.gz  [] mv ${pair\_id}.itdseek\_tandemdup.vcf.gz ${pair\_id}.itdseek\_tandemdup.unfilt.vcf.gz  [] mv ${pair\_id}.itdseek\_tandemdup.pass.vcf ${pair\_id}.itdseek\_tandemdup.vcf  [] bgzip ${pair\_id}.itdseek\_tandemdup.vcf |   8. Base Quality Score Recalibration. Detect systematic errors made by the sequencer when it estimates the quality score of each base call. Build a model of covariation based on the data and a set of known variants, then adjust the base quality scores in the data based on the model.   |  |  | | --- | --- | | executable | gatkrunner.sh gatk/4.1.2.0 samtools/gcc/1.8 | | input | known variants dbsnp, reference fasta, umi consensus bam | | output | Realigned recalibrated bam | | command | [] gatk --java-options "-Xmx32g" BaseRecalibrator -I ${sbam} --known-sites ${index\_path}/dbSnp.gatk4.vcf.gz -R ${reffa} -O ${pair\_id}.recal\_data.table --use-original-qualities  [] gatk --java-options "-Xmx32g" ApplyBQSR -I ${sbam} -R ${reffa} -O ${pair\_id}.final.bam --use-original-qualities -bqsr ${pair\_id}.recal\_data.table  [] samtools index -@ $NPROC ${pair\_id}.final.bam |   9. Gather QC statistics  Gather per sample sequence statistics, create final QC metrics file, generate coverage histogram   |  |  | | --- | --- | | executable | sequenceqc\_alignment\_withumi.pl R/3.2.1-intel git/gcc/v2.12.2 | | input | Alignment stats, library complexity, insert size, coverage stats, ontarget stats, trim report, mapping rate and dedup stats | | output | coverage plot and per sample sequence QC file(includes reference and workflow version). | | command | [] perl sequenceqc\_alignment\_withumi.pl -r ${index\_path} \*.genomecov.txt |   **Alignment Implementation**  The Workflow used in this SOP can be downloaded here: <https://git.biohpc.swmed.edu/brandi.cantarel/clinseq_workflows>  **Usage**  nextflow –C nextflow.config run –w $workdir alignment.nf –design design.txt –capture ${captureBed} –input ${inputdir} –output ${outputdir} --markdups fgbio\_umi & > nextflow\_alignment.log    where $workdir is the folder where the nextflow output files for each step will be deposited, $captureBed is the target capture regions in bed format, $inputdir is the name of the folder where the fastq files are located, the $outputdir is the name of the folder where the output files will be written, design.txt is a tab delimited file that contains the headers: SampleID, FamilyID, FqR1, FqR2. The nextflow\_alignment.log is the log file that tracks nextflow work folders and nextflow errors.  **Tumor Only Variant Detection Procedure**    Figure 2: Tumor Only Variant Detection Overview  1. Index Tumor BAM file   |  |  | | --- | --- | | executable | indexbams.sh samtools/1.6 | | input | tumor BAM file | | output | tumor BAM index file (bai) | | command | [] samtools index -@ $NPROC ${i} |   2. MSI  Calculate Microsatellite Instability Score   |  |  | | --- | --- | | executable | msisensor.sh, msisensor2 | | input | Tumor and normal bam files | | output | msisensor2 output file containing percent msi score | | command | msisensor2 msi -d ${index\_path}/microsatellites.list -t $sbam -o ${pair\_id}.msi |   3. Calling Structural Variants  Identify structural variants using delly   |  |  | | --- | --- | | executable | pindel.sh norm\_annot.sh filter\_pindel.pl samtools/gcc/1.8 snpeff/4.3q samtools/1.6 pindel/0.2.5-intel snpeff/4.3q bedtools/2.26.0 bcftools/1.6 | | input | reference genome fasta, sorted-deduplicated bam, bed files with the gene and exon locations on the reference genome | | output | structural variant vcf files and table of gene annotated structural variants | | command | [] echo -e "${nid}\tcontrol"> samples.tsv  [] echo -e "${tid}\ttumor" >> samples.tsv  [] delly2 call -t BND -o ${pair\_id}.delly\_translocations.bcf -q 30 -g ${reffa} ${sbam} ${normal}  [] delly2 call -t DUP -o ${pair\_id}.delly\_duplications.bcf -q 30 -g ${reffa} ${sbam} ${normal}  [] delly2 call -t INV -o ${pair\_id}.delly\_inversions.bcf -q 30 -g ${reffa} ${sbam} ${normal}  [] delly2 call -t DEL -o ${pair\_id}.delly\_deletion.bcf -q 30 -g ${reffa} ${sbam} ${normal}  [] delly2 call -t INS -o ${pair\_id}.delly\_insertion.bcf -q 30 -g ${reffa} ${sbam} ${normal}  [] bcftools concat -a -O v ${pair\_id}.delly\_duplications.bcf ${pair\_id}.delly\_inversions.bcf ${pair\_id}.delly\_translocations.bcf ${pair\_id}.delly\_deletion.bcf ${pair\_id}.delly\_insertion.bcf | vcf-sort -t temp | bgzip > ${pair\_id}.delly.svar.vcf.gz  [] bash norm\_annot.sh -r ${index\_path} -p ${pair\_id}.delly.sv -v ${pair\_id}.delly.svar.vcf.gz –s  [] perl uniform\_vcf\_gt.pl $pair\_id $vcf  [] bgzip -f ${pair\_id}.uniform.vcf  [] j=${pair\_id}.uniform.vcf.gz  [] tabix -f $j  [] bcftools norm --fasta-ref $reffa -m - -Oz $j -o ${pair\_id}.norm.vcf.gz  [] java -jar $SNPEFF\_HOME/SnpSift.jar filter "( GEN[\*].DP >= 20 )" ${pair\_id}.delly.sv.norm.vcf.gz | java -Xmx10g -jar $SNPEFF\_HOME/snpEff.jar -no-intergenic -lof -c $SNPEFF\_HOME/snpEff.config ${snpeffgeno} - | bgzip > ${pair\_id}.delly.vcf.gz  [] zcat ${pair\_id}.delly.vcf.gz | $SNPEFF\_HOME/scripts/vcfEffOnePerLine.pl |java -jar $SNPEFF\_HOME/SnpSift.jar extractFields - CHROM POS CHR2 END ANN[\*].EFFECT ANN[\*].GENE ANN[\*].BIOTYPE FILTER FORMAT GEN[\*] |grep -E 'gene\_fusion|feature\_fusion' | sort -u > ${pair\_id}.dgf.txt  [ ] mv ${pair\_id}.delly.vcf.gz ${pair\_id}.delly.ori.vcf.gz  [] perl filter\_delly.pl -t $tid -p $pair\_id -i ${pair\_id}.delly.ori.vcf.gz  [] bgzip -f ${pair\_id}.delly.vcf  [] zgrep '#CHROM' ${pair\_id}.delly.vcf.gz > ${pair\_id}.delly.genefusion.txt  [] cat ${pair\_id}.delly.potentialfusion.txt ${pair\_id}.dgf.txt |sort -u >> ${pair\_id}.delly.genefusion.txt |   Identify structural variants using svaba   |  |  | | --- | --- | | executable | pindel.sh norm\_annot.sh filter\_pindel.pl samtools/gcc/1.8 snpeff/4.3q samtools/1.6 pindel/0.2.5-intel snpeff/4.3q bedtools/2.26.0 bcftools/1.6 | | input | reference genome fasta, sorted-deduplicated bam, bed files with the gene and exon locations on the reference genome | | output | structural variant vcf files and table of gene annotated structural variants | | command | [] svaba run -p $NPROC -G ${reffa} -t ${sbam} -n ${normal} -a ${pair\_id}  [] vcf-concat ${pair\_id}.svaba.unfiltered\*sv.vcf | perl -pe 's/\.consensus|\.bam//g' | vcf-sort| bgzip > ${pair\_id}.svaba.unfiltered.sv.vcf.gz  [] bash norm\_annot.sh -r ${index\_path} -p svaba.sv -v ${pair\_id}.svaba.unfiltered.sv.vcf.gz  [] perl uniform\_vcf\_gt.pl $pair\_id $vcf  [] bgzip -f ${pair\_id}.uniform.vcf  [] j=${pair\_id}.uniform.vcf.gz  [] tabix -f $j  [] bcftools norm --fasta-ref $reffa -m - -Oz $j -o ${pair\_id}.norm.vcf.gz  [] java -Xmx10g -jar $SNPEFF\_HOME/snpEff.jar -no-intergenic -lof -c $SNPEFF\_HOME/snpEff.config ${snpeffgeno} svaba.sv.norm.vcf.gz | java -jar $SNPEFF\_HOME/SnpSift.jar filter "( GEN[\*].AO >= 20)" | bgzip > ${pair\_id}.svaba.sv.vcf.gz  [] vcf-concat ${pair\_id}.svaba.unfiltered\*indel.vcf | perl -pe 's/\.consensus|\.bam//g' | vcf-sort | java -jar $SNPEFF\_HOME/SnpSift.jar filter "( SPAN >= 20)" - |bgzip > ${pair\_id}.svaba.indel.vcf.gz  [] norm\_annot.sh -r ${index\_path} -p svaba.indel -v ${pair\_id}.svaba.indel.vcf.gz  [] java -Xmx10g -jar $SNPEFF\_HOME/snpEff.jar -no-intergenic -lof -c $SNPEFF\_HOME/snpEff.config ${snpeffgeno} svaba.indel.norm.vcf.gz | bgzip > ${pair\_id}.svaba.vcf.gz  [] zcat ${pair\_id}.svaba.sv.vcf.gz | $SNPEFF\_HOME/scripts/vcfEffOnePerLine.pl |java -jar $SNPEFF\_HOME/SnpSift.jar extractFields - CHROM POS ALT ID ANN[\*].EFFECT ANN[\*].GENE ANN[\*].BIOTYPE FILTER FORMAT GEN[\*] |grep -E 'gene\_fusion|feature\_fusion' | sort -u > ${pair\_id}.sgf.txt  []mv ${pair\_id}.svaba.vcf.gz ${pair\_id}.svaba.ori.vcf.gz  [] perl filter\_svaba.pl -t $tid -p ${pair\_id} -i ${pair\_id}.svaba.ori.vcf.gz -s ${pair\_id}.svaba.sv.vcf.gz  [] bgzip ${pair\_id}.svaba.vcf  [] zgrep '#CHROM' ${pair\_id}.svaba.sv.vcf.gz > ${pair\_id}.svaba.genefusion.txt  [] cat ${pair\_id}.svaba.potentialfusion.txt ${pair\_id}.sgf.txt | sort -u >> ${pair\_id}.svaba.genefusion.txt |   Identify structural variants using pindel   |  |  | | --- | --- | | executable | pindel.sh norm\_annot.sh filter\_pindel.pl samtools/gcc/1.8 snpeff/4.3q samtools/1.6 pindel/0.2.5-intel snpeff/4.3q bedtools/2.26.0 bcftools/1.6 | | input | reference genome fasta, sorted-deduplicated bam, bed files with the gene and exon locations on the reference genome | | output | structural variant vcf files and table of gene annotated structural variants | | command | [] pindel -T $NPROC -f ${index\_path}/genome.fa -i ${pair\_id}.pindel.config -o ${pair\_id}.pindel\_out --RP  [] pindel2vcf -P ${pair\_id}.pindel\_out -r ${index\_path}/genome.fa -R HG38 -d ${genomefiledate} -v pindel.vcf  [] cat pindel.vcf | java -jar $SNPEFF\_HOME/SnpSift.jar filter " ( GEN[\*].AD[1] >= 10 )" | bgzip > pindel.vcf.gz  [] tabix pindel.vcf.gz  [] bash norm\_annot.sh -r ${index\_path} -p pindel -v pindel.vcf.gz  [] perl uniform\_vcf\_gt.pl $pair\_id $vcf  [] bgzip -f ${pair\_id}.uniform.vcf  [] j=${pair\_id}.uniform.vcf.gz  [] tabix -f $j  [] bcftools norm --fasta-ref $reffa -m - -Oz $j -o ${pair\_id}.norm.vcf.gz  [] perl parse\_pindel.pl ${pair\_id} pindel.norm.vcf.gz  [] java -Xmx10g -jar $SNPEFF\_HOME/snpEff.jar -no-intergenic -lof -c $SNPEFF\_HOME/snpEff.config ${snpeffgeno} ${pair\_id}.indel.vcf |bgzip > ${pair\_id}.pindel\_indel.vcf.gz  [] java -Xmx10g -jar $SNPEFF\_HOME/snpEff.jar -no-intergenic -lof -c $SNPEFF\_HOME/snpEff.config ${snpeffgeno} ${pair\_id}.dup.vcf | bedtools intersect -header -b ${bed} -a stdin | bgzip > ${pair\_id}.pindel\_tandemdup.vcf.gz  [] java -Xmx10g -jar $SNPEFF\_HOME/snpEff.jar -no-intergenic -lof -c $SNPEFF\_HOME/snpEff.config ${snpeffgeno} ${pair\_id}.sv.vcf | bgzip > ${pair\_id}.pindel.sv.vcf.gz  [] perl filter\_pindel.pl -d ${pair\_id}.pindel\_tandemdup.vcf.gz -s ${pair\_id}.pindel.sv.vcf.gz -i ${pair\_id}.pindel\_indel.vcf.gz  [] mv ${pair\_id}.pindel\_tandemdup.vcf.gz ${pair\_id}.pindel\_tandemdup.unfilt.vcf.gz  [] mv ${pair\_id}.pindel\_tandemdup.pass.vcf ${pair\_id}.pindel\_tandemdup.vcf  [] bgzip ${pair\_id}.pindel\_tandemdup.vcf  [] mv ${pair\_id}.pindel\_indel.pass.vcf ${pair\_id}.pindel.vcf  [] bgzip ${pair\_id}.pindel.vcf  [] mv ${pair\_id}.pindel.sv.vcf.gz ${pair\_id}.pindel.sv.unfilt.vcf.gz  [] mv ${pair\_id}.pindel.sv.pass.vcf ${pair\_id}.pindel.sv.vcf  [] bgzip ${pair\_id}.pindel.sv.vcf  [] zgrep '#CHROM' ${pair\_id}.pindel.sv.vcf.gz > ${pair\_id}.pindel.genefusion.txt  [] zcat ${pair\_id}.pindel.sv.vcf.gz | $SNPEFF\_HOME/scripts/vcfEffOnePerLine.pl |java -jar $SNPEFF\_HOME/SnpSift.jar extractFields - CHROM POS CHROM END ANN[\*].EFFECT ANN[\*].GENE ANN[\*].BIOTYPE FILTER FORMAT GEN[\*] |grep -E 'gene\_fusion|feature\_fusion' | sort -u >> ${pair\_id}.pindel.genefusion.txt |   4. Variant Calling and Filtering using Freebayes  Call variants with and base calling quality >= 20, AF >0.1, DP >10, and filter them based on the following cutoffs: DP >= 10.   |  |  | | --- | --- | | executable | germline\_vc.sh uni\_norm\_annot.sh freebayes/gcc/1.2.0 parallel/20150122 bedtools/2.26.0 snpeff/4.3q python/2.7.x-anaconda picard/2.10.3 samtools/gcc/1.8 bcftools/gcc/1.8 vcftools/0.1.14 samtools/1.6 bcftools/1.6 | | Input | reference genome fasta, consensus bam | | output | Freebayes variant call file | | command | [] cut -f 1 ${index\_path}/genomefile.5M.txt | parallel --delay 2 -j $NPROC "freebayes -f ${index\_path}/genome.fa --min-base-quality 20 --min-coverage 10 --min-alternate-fraction 0.01 -C 3 --use-best-n-alleles 3 -r {} ${bamlist} > fb.{}.vcf"  [] vcf-concat fb.\*.vcf | vcf-sort | vcf-annotate -n --fill-type | bcftools norm -c s -f ${reffa} -w 10 -O z -o ${pair\_id}.freebayes.vcf.gz –  [] perl uniform\_vcf\_gt.pl $pair\_id $vcf  [] mv ${vcf} ${pair\_id}.ori.vcf.gz  [] bgzip -f ${pair\_id}.uniform.vcf  [] j=${pair\_id}.uniform.vcf.gz  [] tabix -f $j  [] bcftools norm -m - -Oz $j -o ${pair\_id}.norm.vcf.gz  [] bash annotvcf.sh -p ${pair\_id} -r $index\_path -v ${pair\_id}.norm.vcf.gz  [] vt decompose\_blocksub ${pair\_id}.annot.vcf.gz -p -a -o ${pair\_id}.vcf  [] bgzip -f ${pair\_id}.vcf |   5. Variant Calling and Filtering using GATK  Call variants with HaplotypeCaller identifying FisherStrand bias scores, quality by depth, depth per allele and tandem duplications   |  |  | | --- | --- | | executable | germline\_vc.sh uni\_norm\_annot.sh gatk/4.1.2.0 python/2.7.x-anaconda picard/2.10.3 samtools/gcc/1.8 bcftools/gcc/1.8 bedtools/2.26.0 snpeff/4.3q vcftools/0.1.14 parallel samtools/1.6 bcftools/1.6 | | Input | reference genome fasta, realigned/recalibrated bam, dbsnp vcf | | output | Gatk variant call file | | command | [] gatk --java-options "-Xmx20g" Mutect2 $ponopt -R ${reffa} ${bamlist} --output ${pair\_id}.mutect.vcf -RF AllowAllReadsReadFilter --independent-mates --tmp-dir `pwd`  [] gatk --java-options "-Xmx20g" FilterMutectCalls -R ${reffa} -V ${pair\_id}.mutect.vcf -O ${pair\_id}.mutect.filt.vcf  [] vcf-sort ${pair\_id}.mutect.filt.vcf | vcf-annotate -n --fill-type | java -jar $SNPEFF\_HOME/SnpSift.jar filter -p '(GEN[\*].DP >= 10)' | bgzip > ${pair\_id}.mutect.vcf.gz |   6. Variant Calling and Filtering using Strelka2  Detect variants using Strelka2 and filter with the following cutoffs: QUAL >= 10.   |  |  | | --- | --- | | executable | germline\_vc.sh uni\_norm\_annot.sh python/2.7.x-anaconda picard/2.10.3 bcftools/gcc/1.8 samtools/1.6 bedtools/2.26.0 snpeff/4.3q vcftools/0.1.14 parallel strelka/2.8.3 manta/1.2.0 bcftools/1.6 | | input | reference genome fasta, dbsnp vcf, consensus bam | | output | Strelka variant call file | | command | [] configManta.py --normalBam ${normal} --tumorBam ${tumor} --referenceFasta ${reffa} --runDir manta  [] manta/runWorkflow.py -m local -j 8  configureStrelkaSomaticWorkflow.py --normalBam ${normal} --tumorBam ${tumor} --referenceFasta ${reffa} --targeted --indelCandidates manta/results/variants/candidateSmallIndels.vcf.gz --runDir strelka  [] strelka/runWorkflow.py -m local -j 8  [] vcf-concat strelka/results/variants/\*.vcf.gz | vcf-annotate -n --fill-type -n |vcf-sort |java -jar $SNPEFF\_HOME/SnpSift.jar filter "(GEN[\*].DP >= 10)" | perl -pe "s/TUMOR/${tid}/g" | perl -pe "s/NORMAL/${nid}/g" |bgzip > ${pair\_id}.strelka2.vcf.gz |   7. Variant Calling and Filtering using Platypus  Detect variants using Platypus and filter with the following cutoffs: QUAL >= 10 & QC > 2.   |  |  | | --- | --- | | executable | germline\_vc.sh uni\_norm\_annot.shpython/2.7.x-anaconda picard/2.10.3 samtools/1.6 bcftools/gcc/1.8 samtools/gcc/1.8bedtools/2.26.0 snpeff/4.3q vcftools/0.1.14 parallel platypus/gcc/0.8.1 bcftools/1.6 | | input | reference genome fasta, consensus bam | | output | Platypus variant call file | | command | [] bamlist=`join\_by , \*.bam`  [] Platypus.py callVariants --minMapQual=10 --mergeClusteredVariants=1 --nCPU=$NPROC --bamFiles=${bamlist} --refFile=${reffa} --output=platypus.vcf  [] vcf-sort platypus.vcf |vcf-annotate -n --fill-type -n |bgzip > platypus.vcf.gz  [] tabix platypus.vcf.gz  [] bcftools norm -m - -Oz $j -o ${pair\_id}.norm.vcf.gz  [] bash $baseDir/annotvcf.sh -p ${pair\_id} -r $index\_path -v ${pair\_id}.norm.vcf.gz  /project/shared/bicf\_workflow\_ref/seqprg/vt/vt decompose\_blocksub ${pair\_id}.annot.vcf.gz -p -a -o ${pair\_id}.vcf  [] bgzip -f ${pair\_id}.vcf |   8. Generate union vcf  Integrate result variant lists generated by GATK, SAMtools, Hotspot algorithm, SpeedSeq, and Platypus variant callers. The priority order for ambiguous variants is SAMtools - SpeedSeq - GATK - Platypus - Hotspot. Annotate VCF with publicly available genes   |  |  | | --- | --- | | executable | union.sh bedtools/2.26.0 samtools/1.6 bcftools/1.6 snpeff/4.3q unionvcf.pl vcfsorter.pl | | input | reference genome fasta, GATK, Strelka, Freebayes, Platypus gzipped and annotated vcf | | output | Union vcf | | command | perl unionvcf.pl ${index\_path}/union.header.vcf $list2  perl vcfsorter.pl ${index\_path}/genome.dict int.vcf |bgzip > ${pair\_id}.union.vcf.gz  mv ${subjid}.union.vcf.gz ${subjid}\_${params.projectid}.dna.vcf.gz |   **Tumor Only Variant Detection** **Implementation**  The Workflow used in this SOP can be downloaded here: <https://git.biohpc.swmed.edu/brandi.cantarel/clinseq_workflows>  **Usage**  nextflow -C nextflow.config run -w $workdir tumoronly.nf --design design\_tumor\_only.txt --projectid ${project\_id} --capture ${captureBed} --input $inputdir --output $outnf &> nextflow\_tumoronly.log  where $workdir is the folder where the work files for each step will be deposited, ${captureBed} is the capture region in bed format, ${project\_id} is the run name, the $inputdir is the name of the folder where the bam files are located, the output directory is the name of the folder where the output files will be written, design\_tumor\_only.txt is a tab delimited file with the following headers: SampleID, BAM, OntargetBAM, and the \*nextflow\_tumoronly.log is the log file that tracks nextflow work folders and nextflow errors.  **Somatic Variant Detection Procedure**    Figure 3: Somatic Variant Detection Overview  1. Index BAM files   |  |  | | --- | --- | | executable | samtools/1.6 indexbams.sh | | Input | tumor BAM file, normal BAM file | | Output | tumor BAM index file (bai), normal BAM index file (bai) | | command | [] samtools index -@ $NPROC ${i} |   2. Checkmates and MSI  Determines if sample tumor and normal pairs are from the same individual. Gather quality metrics for somatic pipeline. Calculate miscrosattelite instability score   |  |  | | --- | --- | | executable | python/2.7.x-anaconda sequenceqc\_somatic.pl git/v2.5.3 htslib/gcc/1.8 ncm.py msisensor.sh msisensor2 | | input | tumor BAM file, normal BAM file, reference genome FASTA file, ngs checkmate bed file, ncm.conf, dbSnp vcf file, cosmic vcf file | | output | NGSCheckmate output file with similarity score for each sample pair, qc metrics file, msisensor2 output file | | Command | [] python /project/shared/bicf\_workflow\_ref/seqprg/NGSCheckMate/ncm.py -B -d ./ -bed ${index\_path}/NGSCheckMate.bed -O ./ -N ${pid}  [] perl sequenceqc\_somatic.pl -r ${index\_path} -i ${pid}\_all.txt -o ${pid}\_${projectid}.sequence.stats.txt  msisensor2 msi -d ${index\_path}/microsatellites.list -n $normal -t $sbam -o ${pair\_id}.msi |   3. Calling Structural Variants  Identify structural variants using delly   |  |  | | --- | --- | | executable | svcalling.sh norm\_annot.sh filter\_delly.pl samtools/gcc/1.8 bcftools/gcc/1.8 snpeff/4.3q | | input | reference genome FASTA, genome dict file, sorted BAM file, | | output | structural variant vcf files and table of gene annotated structural variants | | command | [] echo -e "${nid}\tcontrol"> samples.tsv  [] echo -e "${tid}\ttumor" >> samples.tsv  [] delly2 call -t BND -o ${pair\_id}.delly\_translocations.bcf -q 30 -g ${reffa} ${sbam} ${normal}  [] delly2 call -t DUP -o ${pair\_id}.delly\_duplications.bcf -q 30 -g ${reffa} ${sbam} ${normal}  [] delly2 call -t INV -o ${pair\_id}.delly\_inversions.bcf -q 30 -g ${reffa} ${sbam} ${normal}  [] delly2 call -t DEL -o ${pair\_id}.delly\_deletion.bcf -q 30 -g ${reffa} ${sbam} ${normal}  [] delly2 call -t INS -o ${pair\_id}.delly\_insertion.bcf -q 30 -g ${reffa} ${sbam} ${normal}  [] bcftools concat -a -O v ${pair\_id}.delly\_duplications.bcf ${pair\_id}.delly\_inversions.bcf ${pair\_id}.delly\_translocations.bcf ${pair\_id}.delly\_deletion.bcf ${pair\_id}.delly\_insertion.bcf | vcf-sort -t temp | bgzip > ${pair\_id}.delly.svar.vcf.gz  [] bash norm\_annot.sh -r ${index\_path} -p ${pair\_id}.delly.sv -v ${pair\_id}.delly.svar.vcf.gz –s  [] perl uniform\_vcf\_gt.pl $pair\_id $vcf  [] bgzip -f ${pair\_id}.uniform.vcf  [] j=${pair\_id}.uniform.vcf.gz  [] tabix -f $j  [] bcftools norm --fasta-ref $reffa -m - -Oz $j -o ${pair\_id}.norm.vcf.gz  [] java -jar $SNPEFF\_HOME/SnpSift.jar filter "( GEN[\*].DP >= 20 )" ${pair\_id}.delly.sv.norm.vcf.gz | java -Xmx10g -jar $SNPEFF\_HOME/snpEff.jar -no-intergenic -lof -c $SNPEFF\_HOME/snpEff.config ${snpeffgeno} - | bgzip > ${pair\_id}.delly.vcf.gz  [] zcat ${pair\_id}.delly.vcf.gz | $SNPEFF\_HOME/scripts/vcfEffOnePerLine.pl |java -jar $SNPEFF\_HOME/SnpSift.jar extractFields - CHROM POS CHR2 END ANN[\*].EFFECT ANN[\*].GENE ANN[\*].BIOTYPE FILTER FORMAT GEN[\*] |grep -E 'gene\_fusion|feature\_fusion' | sort -u > ${pair\_id}.dgf.txt  [ ] mv ${pair\_id}.delly.vcf.gz ${pair\_id}.delly.ori.vcf.gz  [] perl filter\_delly.pl -t $tid -p $pair\_id -i ${pair\_id}.delly.ori.vcf.gz  [] bgzip -f ${pair\_id}.delly.vcf  [] zgrep '#CHROM' ${pair\_id}.delly.vcf.gz > ${pair\_id}.delly.genefusion.txt  [] cat ${pair\_id}.delly.potentialfusion.txt ${pair\_id}.dgf.txt |sort -u >> ${pair\_id}.delly.genefusion.txt |   Identify structural variants using svaba   |  |  | | --- | --- | | executable | svcalling.sh norm\_annot.sh filter\_delly.pl samtools/gcc/1.8 bcftools/gcc/1.8 snpeff/4.3q vcftools/0.1.14 | | input | reference genome FASTA, genome dict file, sorted BAM file, | | output | structural variant vcf files and table of gene annotated structural variants | | command | [] svaba run -p $NPROC -G ${reffa} -t ${sbam} -n ${normal} -a ${pair\_id}  [] vcf-concat ${pair\_id}.svaba.unfiltered\*sv.vcf | perl -pe 's/\.consensus|\.bam//g' | vcf-sort| bgzip > ${pair\_id}.svaba.unfiltered.sv.vcf.gz  [] bash norm\_annot.sh -r ${index\_path} -p svaba.sv -v ${pair\_id}.svaba.unfiltered.sv.vcf.gz  [] perl uniform\_vcf\_gt.pl $pair\_id $vcf  [] bgzip -f ${pair\_id}.uniform.vcf  [] j=${pair\_id}.uniform.vcf.gz  [] tabix -f $j  [] bcftools norm --fasta-ref $reffa -m - -Oz $j -o ${pair\_id}.norm.vcf.gz  [] java -Xmx10g -jar $SNPEFF\_HOME/snpEff.jar -no-intergenic -lof -c $SNPEFF\_HOME/snpEff.config ${snpeffgeno} svaba.sv.norm.vcf.gz | java -jar $SNPEFF\_HOME/SnpSift.jar filter "( GEN[\*].AO >= 20)" | bgzip > ${pair\_id}.svaba.sv.vcf.gz  [] vcf-concat ${pair\_id}.svaba.unfiltered\*indel.vcf | perl -pe 's/\.consensus|\.bam//g' | vcf-sort | java -jar $SNPEFF\_HOME/SnpSift.jar filter "( SPAN >= 20)" - |bgzip > ${pair\_id}.svaba.indel.vcf.gz  [] norm\_annot.sh -r ${index\_path} -p svaba.indel -v ${pair\_id}.svaba.indel.vcf.gz  [] java -Xmx10g -jar $SNPEFF\_HOME/snpEff.jar -no-intergenic -lof -c $SNPEFF\_HOME/snpEff.config ${snpeffgeno} svaba.indel.norm.vcf.gz | bgzip > ${pair\_id}.svaba.vcf.gz  [] zcat ${pair\_id}.svaba.sv.vcf.gz | $SNPEFF\_HOME/scripts/vcfEffOnePerLine.pl |java -jar $SNPEFF\_HOME/SnpSift.jar extractFields - CHROM POS ALT ID ANN[\*].EFFECT ANN[\*].GENE ANN[\*].BIOTYPE FILTER FORMAT GEN[\*] |grep -E 'gene\_fusion|feature\_fusion' | sort -u > ${pair\_id}.sgf.txt  []mv ${pair\_id}.svaba.vcf.gz ${pair\_id}.svaba.ori.vcf.gz  [] perl filter\_svaba.pl -t $tid -p ${pair\_id} -i ${pair\_id}.svaba.ori.vcf.gz -s ${pair\_id}.svaba.sv.vcf.gz  [] bgzip ${pair\_id}.svaba.vcf  [] zgrep '#CHROM' ${pair\_id}.svaba.sv.vcf.gz > ${pair\_id}.svaba.genefusion.txt  [] cat ${pair\_id}.svaba.potentialfusion.txt ${pair\_id}.sgf.txt | sort -u >> ${pair\_id}.svaba.genefusion.txt |   Identify structural variants using pindel   |  |  | | --- | --- | | executable | pindel.sh norm\_annot.sh filter\_pindel.sh parse\_pindel.pl samtools/gcc/1.8 snpeff/4.3q samtools/1.6 pindel/0.2.5-intel snpeff/4.3q bedtools/2.26.0 bcftools/1.6 | | input | reference genome fasta, sorted-deduplicated bam, bed files with the gene and exon locations on the reference genome | | output | structural variant vcf files and table of gene annotated structural variants | | command | [] pindel -T $NPROC -f ${index\_path}/genome.fa -i ${pair\_id}.pindel.config -o ${pair\_id}.pindel\_out --RP  [] pindel2vcf -P ${pair\_id}.pindel\_out -r ${index\_path}/genome.fa -R HG38 -d ${genomefiledate} -v pindel.vcf  [] cat pindel.vcf | java -jar $SNPEFF\_HOME/SnpSift.jar filter " ( GEN[\*].AD[1] >= 10 )" | bgzip > pindel.vcf.gz  [] tabix pindel.vcf.gz  [] bash norm\_annot.sh -r ${index\_path} -p pindel -v pindel.vcf.gz  [] perl uniform\_vcf\_gt.pl $pair\_id $vcf  [] bgzip -f ${pair\_id}.uniform.vcf  [] j=${pair\_id}.uniform.vcf.gz  [] tabix -f $j  [] bcftools norm --fasta-ref $reffa -m - -Oz $j -o ${pair\_id}.norm.vcf.gz  [] perl parse\_pindel.pl ${pair\_id} pindel.norm.vcf.gz  [] java -Xmx10g -jar $SNPEFF\_HOME/snpEff.jar -no-intergenic -lof -c $SNPEFF\_HOME/snpEff.config ${snpeffgeno} ${pair\_id}.indel.vcf |bgzip > ${pair\_id}.pindel\_indel.vcf.gz  [] java -Xmx10g -jar $SNPEFF\_HOME/snpEff.jar -no-intergenic -lof -c $SNPEFF\_HOME/snpEff.config ${snpeffgeno} ${pair\_id}.dup.vcf | bedtools intersect -header -b ${bed} -a stdin | bgzip > ${pair\_id}.pindel\_tandemdup.vcf.gz  [] java -Xmx10g -jar $SNPEFF\_HOME/snpEff.jar -no-intergenic -lof -c $SNPEFF\_HOME/snpEff.config ${snpeffgeno} ${pair\_id}.sv.vcf | bgzip > ${pair\_id}.pindel.sv.vcf.gz  [] perl filter\_pindel.pl -d ${pair\_id}.pindel\_tandemdup.vcf.gz -s ${pair\_id}.pindel.sv.vcf.gz -i ${pair\_id}.pindel\_indel.vcf.gz  [] mv ${pair\_id}.pindel\_tandemdup.vcf.gz ${pair\_id}.pindel\_tandemdup.unfilt.vcf.gz  [] mv ${pair\_id}.pindel\_tandemdup.pass.vcf ${pair\_id}.pindel\_tandemdup.vcf  [] bgzip ${pair\_id}.pindel\_tandemdup.vcf  [] mv ${pair\_id}.pindel\_indel.pass.vcf ${pair\_id}.pindel.vcf  [] bgzip ${pair\_id}.pindel.vcf  [] mv ${pair\_id}.pindel.sv.vcf.gz ${pair\_id}.pindel.sv.unfilt.vcf.gz  [] mv ${pair\_id}.pindel.sv.pass.vcf ${pair\_id}.pindel.sv.vcf  [] bgzip ${pair\_id}.pindel.sv.vcf  [] zgrep '#CHROM' ${pair\_id}.pindel.sv.vcf.gz > ${pair\_id}.pindel.genefusion.txt  [] zcat ${pair\_id}.pindel.sv.vcf.gz | $SNPEFF\_HOME/scripts/vcfEffOnePerLine.pl |java -jar $SNPEFF\_HOME/SnpSift.jar extractFields - CHROM POS CHROM END ANN[\*].EFFECT ANN[\*].GENE ANN[\*].BIOTYPE FILTER FORMAT GEN[\*] |grep -E 'gene\_fusion|feature\_fusion' | sort -u >> ${pair\_id}.pindel.genefusion.txt |   4. Variant Calling and Filtering using Freebayes  Call variants with and base calling quality >= 20, AF >0.1, DP >10, and filter them based on the following cutoffs: DP >= 10.   |  |  | | --- | --- | | executable | germline\_vc.sh uni\_norm\_annot.sh freebayes/gcc/1.2.0 parallel/20150122 bedtools/2.26.0 samtools/1.6 bcftools/1.6 snpeff/4.3q python/2.7.x-anaconda picard/2.10.3 samtools/gcc/1.8 bcftools/gcc/1.8 vcftools/0.1.14 | | Input | reference genome fasta, consensus bam | | output | Freebayes variant call file | | command | [] cut -f 1 ${index\_path}/genomefile.5M.txt | parallel --delay 2 -j $NPROC "freebayes -f ${index\_path}/genome.fa --min-base-quality 20 --min-coverage 10 --min-alternate-fraction 0.01 -C 3 --use-best-n-alleles 3 -r {} ${bamlist} > fb.{}.vcf"  [] vcf-concat fb.\*.vcf | vcf-sort | vcf-annotate -n --fill-type | bcftools norm -c s -f ${reffa} -w 10 -O z -o ${pair\_id}.freebayes.vcf.gz –  [] perl uniform\_vcf\_gt.pl $pair\_id $vcf  [] mv ${vcf} ${pair\_id}.ori.vcf.gz  [] bgzip -f ${pair\_id}.uniform.vcf  [] j=${pair\_id}.uniform.vcf.gz  [] tabix -f $j  [] bcftools norm -m - -Oz $j -o ${pair\_id}.norm.vcf.gz  [] bash annotvcf.sh -p ${pair\_id} -r $index\_path -v ${pair\_id}.norm.vcf.gz  [] vt decompose\_blocksub ${pair\_id}.annot.vcf.gz -p -a -o ${pair\_id}.vcf  [] bgzip -f ${pair\_id}.vcf |   5. Variant Calling and Filtering using Platypus  Detect variants using Platypus and filter with the following cutoffs: QUAL >= 10 & QC > 2.   |  |  | | --- | --- | | executable | germline\_vc.sh uni\_norm\_annot.sh python/2.7.x-anaconda picard/2.10.3 samtools/1.6 bedtools/2.26.0 snpeff/4.3q vcftools/0.1.14 parallel platypus/gcc/0.8.1 | | Input | reference genome fasta, consensus bam | | output | Platypus variant call file | | command | [] bamlist=`join\_by , \*.bam`  [] Platypus.py callVariants --minMapQual=10 --mergeClusteredVariants=1 --nCPU=$NPROC --bamFiles=${bamlist} --refFile=${reffa} --output=platypus.vcf  [] vcf-sort platypus.vcf |vcf-annotate -n --fill-type -n |bgzip > platypus.vcf.gz  [] tabix platypus.vcf.gz  [] bcftools norm -m - -Oz $j -o ${pair\_id}.norm.vcf.gz  [] perl uniform\_vcf\_gt.pl $pair\_id $vcf  [] mv ${vcf} ${pair\_id}.ori.vcf.gz  [] bgzip -f ${pair\_id}.uniform.vcf  [] j=${pair\_id}.uniform.vcf.gz  [] tabix -f $j  [] bcftools norm --fasta-ref $reffa -m - -Oz $j -o ${pair\_id}.norm.vcf.gz  [] bash annotvcf.sh -p ${pair\_id} -r $index\_path -v ${pair\_id}.norm.vcf.gz  [] vt decompose\_blocksub ${pair\_id}.annot.vcf.gz -p -a -o ${pair\_id}.vcf  [] bgzip -f ${pair\_id}.vcf |   6. Somatic variant calling using MuTect2  Detect somatic variants with confidence quality >= 10 using MuTect. The variants are filtered based on the fisher strand value (<= 60) and read depth (>= 10).   |  |  | | --- | --- | | executable | somatic\_vc.sh uni\_norm\_annot.sh gatk/4.1.4.0 singularity/2.6.1 picard/2.10.3 htslib/gcc/1.8 bedtools/2.26.0 samtools/gcc/1.8 bcftools/1.6 snpeff/4.3q vcftools/0.1.14 | | Input | tumor BAM file, normal BAM file, reference genome FASTA file, dbSNP VCF file, COSMIC VCF file | | output | somatic variant calls in VCF | | command | [] java -XX:ParallelGCThreads=$NPROC -Djava.io.tmpdir=./ -Xmx16g -jar $PICARD/picard.jar CollectSequencingArtifactMetrics I=${tumor} O=artifact\_metrics.txt R=${reffa}  [] gatk --java-options "-Xmx20g" Mutect2 $ponopt --independent-mates -RF AllowAllReadsReadFilter -R ${reffa} -I ${tumor} -tumor ${tid} -I ${normal} -normal ${nid} --output ${tid}.mutect.vcf  [] gatk --java-options "-Xmx20g" FilterMutectCalls -R ${reffa} -V ${tid}.mutect.vcf -O ${tid}.mutect.filt.vcf  [] vcf-sort ${tid}.mutect.filt.vcf | vcf-annotate -n --fill-type | java -jar $SNPEFF\_HOME/SnpSift.jar filter -p '(GEN[\*].DP >= 10)' | bgzip > ${pair\_id}.mutect.vcf.gz  [] perl uniform\_vcf\_gt.pl $pair\_id $vcf  [] mv ${vcf} ${pair\_id}.ori.vcf.gz  [] bgzip -f ${pair\_id}.uniform.vcf  [] j=${pair\_id}.uniform.vcf.gz  [] tabix -f $j  [] bcftools norm --fasta-ref $reffa -m - -Oz $j -o ${pair\_id}.norm.vcf.gz  [] bash annotvcf.sh -p ${pair\_id} -r $index\_path -v ${pair\_id}.norm.vcf.gz -g $snpeffgeno  [] vt decompose\_blocksub ${pair\_id}.annot.vcf.gz -p -a -o ${pair\_id}.vcf  [] bgzip -f ${pair\_id}.vcf |   7. Somatic variant calling using Strelka2  Detect somatic variants using Strelka2. The variants are filtered based on the target regions and read depth (>= 10).   |  |  | | --- | --- | | executable | somatic\_vc.sh uni\_norm\_annot.sh strelka/2.9.10 manta/1.3.1 samtools/gcc/1.8 snpeff/4.3q vcftools/0.1.14 htslib/gcc/1.8 bedtools/2.26.0 samtools/1.6 bcftools/1.6 snpeff/4.3q | | Input | tumor BAM file, normal BAM file, reference genome FASTA file, target regions | | output | somatic variant calls in VCF | | command | [] mkdir manta strelka  [] configManta.py --normalBam ${mnormal} --tumorBam ${mtumor} --referenceFasta ${reffa} --runDir manta  [] manta/runWorkflow.py -m local -j 8  [] configureStrelkaSomaticWorkflow.py --normalBam ${normal} --tumorBam ${tumor} --referenceFasta ${reffa} --targeted --indelCandidates manta/results/variants/candidateSmallIndels.vcf.gz --runDir strelka  [] strelka/runWorkflow.py -m local -j 8  [] vcf-concat strelka/results/variants/\*.vcf.gz | vcf-annotate -n --fill-type -n |vcf-sort |java -jar $SNPEFF\_HOME/SnpSift.jar filter "(GEN[\*].DP >= 10)" | perl -pe "s/TUMOR/${tid}/g" | perl -pe "s/NORMAL/${nid}/g" |bgzip > ${pair\_id}.strelka2.vcf.gz  [] perl uniform\_vcf\_gt.pl $pair\_id $vcf  [] mv ${vcf} ${pair\_id}.ori.vcf.gz  [] bgzip -f ${pair\_id}.uniform.vcf  [] j=${pair\_id}.uniform.vcf.gz  [] tabix -f $j  [] bcftools norm --fasta-ref $reffa -m - -Oz $j -o ${pair\_id}.norm.vcf.gz  [] bash annotvcf.sh -p ${pair\_id} -r $index\_path -v ${pair\_id}.norm.vcf.gz -g $snpeffgeno  [] vt decompose\_blocksub ${pair\_id}.annot.vcf.gz -p -a -o ${pair\_id}.vcf  [] bgzip -f ${pair\_id}.vcf |   8. Somatic variant calling using Shimmer  Detect somatic variants with the quality score >= 25 using Shimmer. The variants are filtered based on the target regions and read depth (>= 10).   |  |  | | --- | --- | | executable | somatic\_vc.sh uni\_norm\_annot.sh snpeff/4.3q shimmer/0.1.1 samtools/gcc/1.8 vcftools/0.1.14 bedtools/2.26.0 samtools/1.6 bcftools/1.6 snpeff/4.3q htslib/gcc/1.8 | | Input | tumor BAM file, normal BAM file, reference genome FASTA file, target regions | | output | somatic variant calls in VCF | | command | [] shimmer.pl --minqual 25 --ref ${reffa} ${normal} ${tumor} --outdir shimmer 2> shimmer.err  [] perl add\_readct\_shimmer.pl  [] vcf-annotate -n --fill-type shimmer/somatic\_diffs.readct.vcf | java -jar $SNPEFF\_HOME/SnpSift.jar filter '(GEN[\*].DP >= 10)' | perl -pe "s/TUMOR/${tid}/" | perl -pe "s/NORMAL/${nid}/g" | bgzip > ${pair\_id}.shimmer.vcf.gz  [] perl uniform\_vcf\_gt.pl $pair\_id $vcf  [] mv ${vcf} ${pair\_id}.ori.vcf.gz  [] bgzip -f ${pair\_id}.uniform.vcf  [] j=${pair\_id}.uniform.vcf.gz  [] tabix -f $j  [] bcftools norm --fasta-ref $reffa -m - -Oz $j -o ${pair\_id}.norm.vcf.gz  [] bash annotvcf.sh -p ${pair\_id} -r $index\_path -v ${pair\_id}.norm.vcf.gz -g $snpeffgeno  [] vt decompose\_blocksub ${pair\_id}.annot.vcf.gz -p -a -o ${pair\_id}.vcf  [] bgzip -f ${pair\_id}.vcf |   9. Generate union/Annot VCF  Integrate result somatic variant lists generated by Freebayes, Platypus, Mutect, Strelka and Shimmer.   |  |  | | --- | --- | | executable | union.sh unionvcf.pl vcfsorter.pl python/2.7.x-anaconda bedtools/2.26.0 samtools/1.6 bcftools/1.6 snpeff/4.3q | | Input | VCF files generated by SpeedSeq, MuTect, VarScan, Shimmer, and Virmid, reference genome fasta | | output | union variant calls in VCF, annotated vcf | | command | [] perl unionvcf.pl ${index\_path}/union.header.vcf $list2  [] perl vcfsorter.pl ${index\_path}/genome.dict int.vcf |bgzip > ${pair\_id}.union.vcf.gz |   **Somatic Variant Detection Implementation**  The Workflow used in this SOP can be downloaded here: <https://git.biohpc.swmed.edu/brandi.cantarel/clinseq_workflows>  **Usage**  nextflow -C nextflow.config run -w $workdir somatic.nf --design design\_tumor\_normal.txt --projectid ${project\_id} --input $inputdir --output $outputdir &> nextflow\_somatic.log  where $inputdir is the name of the folder where the BAM files are located, ${project\_id} is the run name,$workdir is the folder where the work files will be run, $outputdir is the name of the folder where the output files will be written. design\_tumor\_normal.txt is a tab delimited file that contains the headers: PairID, VcfID, TumorID, NormalID, TumorBAM, NormalBAM, TumorCBAM, NormalCBAM, TumorGATKBAM, NormalGATKBAM  **CALCULATIONS**  N/A  **INTERPRETATION/RESULTS/ALERT VALUES**  N/A  **REFERENCE INTERVALS (NORMAL REFERENCE RANGE)**  N/A  **ANALYTICAL MEASURING RANGE (LINEARITY)**  N/A  **RESULT REPORTING CRITERIA**  N/A | | |
| **Definitions**   |  |  | | --- | --- | | Term | Definition | | Indel | Insertion and Deletions | | SNP | Single Nucleotide Polymorphism | | | |
| **Applicable Forms**   |  |  | | --- | --- | | Document ID# | Title of Document | | **3011** | DNA\_Analysis | |  |  | |  |  |   **Related Documents**   |  |  | | --- | --- | | Document ID# | Title of Document | | **3012** | Overview\_Bioinformatics\_SOP | |  |  | |  |  | | | |
|  | | |
| **Review, Revision, and Approval History**   |  |  |  |  | | --- | --- | --- | --- | | **History of Document** | | | | | Version | Date | Description of Change | Authored/Revised by | | 1 | 2017/06 | Policy Tech upload, SOP update for UMI | Brandi Cantarel, Erika Villa | | 2 | 2018/08 | Policy Tech upload, SOP update to improve QC calculations | Brandi Cantarel, Erika Villa | | 3 | 2019/07 | Freebayes, starfusion update, Pipeline update | Brandi Cantarel, Erika Villa | | 4 | 2020/04 | Policy Tech upload, adding structural variant tools, changes made for idt bait panels | Brandi Cantarel, Erika Villa | | |
|  | |
| **Contact for Further Information** | |
|  | | |
|  | | |
|  | | |

**Title:** { Title } **Version:** { Version Number } **Status:** { Status }

**Writer(s)**: { Writers > All Users > Name & Title }

**Approver(s)**: { Approvers > All Users > Name & Title } **Approved Date**: { Date Approved }

**Effective Date**: { Publication Date }

**Next Review Date**: { Next Periodic Review Date }