**Variant Calling and Quality Filtration on High-Throughput Sequence (HTS) data**

**Aim**

The aim of this standard operating procedure (SOP) is to provide guidance on variant calling and quality filtration for whole exome sequencing data from HiSeq 2000 sequencer. The variant file generated from this procedure is the initial variant file used for further evaluation.

The analysis is performed by the bioinformatician.

**Tools**

The variant calling and quality filtration are performed by the bioinformatics tools: GATK (version v2.17). GATK is a tool kit to handle high-throughput sequencing data for variant calling.

**Input**

The inputs are refined BAM file (Input\_BAM\_file) and its index file, which are generated from refinement alignment step. The BAM file should be sorted by the chromosome (1~22, X, Y, M) and genomic coordinates, and contain read group information.

**Procedure**

**1, Preparing variant calling target regions**

* add 50 bp on each end of the probe/bait region from capture kit;
* merge overlapping regions;
* reformat region into the following format: chromosome:start-end, the chromosome should be represented by 1, 2, 3, X, Y, MT etc. instead of chr1;
* Sorted regions by the chromosome (1~22, X, Y, M) and genomic coordinates;
* Copy the content of bundle/1.5/b37/human\_g1k\_v37\_decoy.dict file at the beginning of the file.

**2, Variant calling**

java -Xmx4g -jar GenomeAnalysisTK.jar

-R bundle/1.5/b37/human\_g1k\_v37\_decoy.fasta

-T UnifiedGenotyper

-I all.realigned.markDup.baseQreCali.bam

--dbsnp bundle/1.5/b37/dbsnp\_135.b37.vcf

-o all.raw.vcf

-stand\_call\_conf 50

-stand\_emit\_conf 10

-L /data.odin/common/captureTechnologies/agilent/SureSelect\_50Mb\_exome\_v11\_hg19/30\_generatedData/agilent.SureSelect\_50Mb\_exome\_v11\_hg19.50bpExt\_merged.b37.interval\_list

-nt 3

--downsampling\_type ALL\_READS

-glm BOTH

-dcov 200

2>all.raw.vcf.err > all.raw.vcf.std

java -Xmx2g -jar GenomeAnalysisTK.jar

-R bundle/1.5/b37/human\_g1k\_v37\_decoy.fasta

-T SelectVariants

--variant all.raw.vcf

-o snp.raw.vcf

-selectType SNP

-nt 3

2>snp.raw.vcf.err > snp.raw.vcf.std

java -Xmx2g -jar GenomeAnalysisTK.jar

-R bundle/1.5/b37/human\_g1k\_v37\_decoy.fasta

-T SelectVariants

--variant all.raw.vcf

-o indel.raw.vcf

-selectType INDEL

-nt 3

2>indel.raw.vcf.err > indel.raw.vcf.std

**3, SNP Variant Quality Score Recalibration (SNP VQSR)**

java -Xmx4g -jar GenomeAnalysisTK.jar

-T VariantRecalibrator

-R bundle/1.5/b37/human\_g1k\_v37\_decoy.fasta

-input snp.raw.vcf

-resource:hapmap,known=false,training=true,truth=true,prior=15.0 bundle/1.5/b37/hapmap\_3.3.b37.sites.vcf

-resource:omni,known=false,training=true,truth=false,prior=12.0 bundle/1.5/b37/1000G\_omni2.5.b37.sites.vcf

-resource:dbsnp,known=true,training=false,truth=false,prior=6.0 bundle/1.5/b37/dbsnp\_135.b37.vcf

-an QD -an HaplotypeScore -an MQRankSum -an ReadPosRankSum -an FS -an MQ

-recalFile snp.vqsr.output.recal

-tranchesFile snp.vqsr.output.tranches

-rscriptFile snp.vqsr.output.R

--maxGaussians 6

-mode SNP

--target\_titv 3.2

2>snp.VariantRecalibrator.err > snp.variantRecalibrator.std

java -Xmx3g -jar GenomeAnalysisTK.jar

-T ApplyRecalibration

-R bundle/1.5/b37/human\_g1k\_v37\_decoy.fasta

-input snp.raw.vcf

--ts\_filter\_level 99.0

-tranchesFile snp.vqsr.output.tranches

-recalFile snp.vqsr.output.recal

-o snp.recalibrated.filtered.vcf

> snp.recalibrated.filtered.vcf.std

**4, Indel hard filtering**

java -Xmx2g -jar GenomeAnalysisTK.jar

-R bundle/1.5/b37/human\_g1k\_v37\_decoy.fasta

-T VariantFiltration

-o indel.hardFiltered.vcf

--variant indel.raw.vcf

--filterExpression "QD < 2.0"

--filterName QDFilter

--filterExpression "ReadPosRankSum < -20.0"

--filterName ReadPosFilter

--filterExpression "FS > 200.0"

--filterName FSFilter

2>indel.hardFiltered.vcf.err >indel.hardFiltered.vcf.std

**5, Merge SNP and indel filtration VCF files**

java -Xmx2g -jar GenomeAnalysisTK.jar

-R bundle/1.5/b37/human\_g1k\_v37\_decoy.fasta

-T CombineVariants

--variant snp.recalibrated.filtered.vcf

--variant indel.hardFiltered.vcf

-o all.filter.vcf

2>all.filter.vcf.err > all.filter.vcf.std

(See attachments for detail about the options in GATK).

**Output**

1, The VCF file and its index file (all.filter.vcf.idx) from the step 4 (Merge SNP and indel filtration). That means all the variants have been annotated with whether they have good enough quality (pass) or not.

The Variant Call Format (**VCF**) is a specification for storing gene sequence variations. The format has been developed with the advent of large-scale genotyping and gene sequencing projects, such as the 1000 Genomes Project. VCF is a text file format (most likely stored in a compressed manner). It contains meta-information lines, a header line, and then data lines each containing information about a position in the genome. The VCF file could contain variants called from one single sample or multiple samples.

The meta-information lines are started the ## string and must be key=value pairs within them. A single 'fileformat' field is always required, must be the first line in the file, and details the VCF format version number. For example, for VCF version 4.1, this line should read:

##fileformat=VCFv4.1

It is strongly encouraged that information lines describing the INFO, FILTER and FORMAT entries used in the body of the VCF file be included in the meta-information section.

The header line names the 8 fixed, mandatory columns. These columns are as follows:

1. #CHROM – chromosome
2. POS - position
3. ID – semi-colon separated list of unique identifiers where available.
4. REF – reference base(s)
5. ALT – comma separated list of alternate non-reference alleles called on at least one of the samples.
6. QUAL – phred-scaled quality score for the assertion made in ALT. The quality score indicates the probability of the called variant is wrong. The score is between zero and infinity. The larger, the variant call is more probably right.
7. FILTER – PASS if this position has passed all filters. Otherwise if the site has not passed all filters, a semicolon-separated list of codes for filters that fail.
8. INFO – additional information

The header line is tab-delimited.

If genotype information is present, then the same types of data must be present for all samples. First a FORMAT field is given specifying the data types and order (colon-separated alphanumeric String). This is followed by one field per sample, with the colon-separated data in this field corresponding to the types specified in the format. The first sub-field must always be the genotype (GT) if it is present.

2, err and Info Outputs

Both err\* and \*Info\* files are logging files for each step. if all processes running successfully:

* All GATK err files should be empty;
* In all GATK info files, there should be a line says “Total runtime …”
* In all Picard err files, there should be a line says “Elapsed time …”

(See attachments for detail about VCF specifications).

**Variation**

The whole procedure needs to be rerun from the first failed step.

Storage

See “SOP for storage and security of high-throughput sequencing data”.

**Reference**

1, McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, Kernytsky A, Garimella K, Altshuler D, Gabriel S, Daly M, DePristo MA (2010). **The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data.** *Genome Res.* 20:1297-303.

2, DePristo M, Banks E, Poplin R, Garimella K, Maguire J, Hartl C, Philippakis A, del Angel G, Rivas MA, Hanna M, MaKenna A, Fennell T, Kernytsky A, Sivachenko A, Cibulskis K, Gabriel S, Altshuler D and Daly, M (2011). **A framework for variation discovery and genotyping using next-generation DNA sequencing data.** *Nature Genetics.* 43:491-498.

Appendix

1, GATK general options

2, GATK options for UnifiedGenotyper

3, GATK options for SelectVariants

4, GATK options for VariantRecalibrator

5, GATK options for ApplyRecalibration

6, GATK options for VariantFiltration

7, GATK options for CombineVariants

8, VCF format specification