**Clinical Genome Analytics**

**Workflow for Whole Exome Analysis**

**Prepared by**

**Anuj Srivastava and Grace A Stafford**

**Note** most of these applications will name the output based on the input. In this document, the example file names will be based on this convention. Specific input for each usage example are bracketed by < > ; parameters will be set and remain unchanged.

**Assumption**: job initiated in same directory as data files. If not, include path to data. **NOTE** that subsequent commands assume output path of prior commands will not change.

**Step1: NGS QC**

**Module Requirement:**

module load NGSQCToolkit/2.3

* 1. **Quality Statistics generation and bad read filtering**

**Illumina sequencing; CASAVA 1.8+; paired end reads; human data**

**Usage:**

IlluQC\_PRLL.pl -pe <read1.fastq> <read2.fastq> <adaptor\_file> <5> –s 30 -c 12 -o filter

**Input:**

* Forward paired end file [required]
* Reverse paired end file [required]
* Adaptor\_database file or application library [required]
* FASTQ encoding [required]

**Output:**

Folder named “filter” created here which has the following files

* read1.fastq\_filtered (here)
* read2.fastq\_filtered (here)
* \*\_unPaired\_HQReads
* \*\_stat
* \*.html,
* 6 plots for each fastq file\*\_avgQual.png, \*\_baseCompostion.png, \*\_QualRangePerBase.png, \*\_ gcDistribution.png, \*\_ qualDistribution.png, \*\_ QualRangePerBase.png,
* \*\_summary.png (summary plot)

**Parameters Description:**

* -pe: Forward reverse paired end file
* Adaptor\_file: Should be replaced by known adaptor sequence used in particular type of sequencing
* FASTQ quality value variants: 5 stands for = Illumina (1.8+) (Phred+33, 33 to 7)
* -s: -cutOffQualScore (The cut-off value for PHRED quality score for high-quality filtering)
* -c: Number of CPUs to be used
* -o: Output will be stored in the given folder

**1.2 Quality based trimming of filtered files**

**Usage:**

TrimmingReads.pl –i <filter/read1.fastq\_filtered> -irev <filter/read2.fastq\_filtered> –q 30

**Input:**

* Forward paired end file with paired end intact after filtering [required]
* Reverse paired end file with paired end intact after filtering [required]

**Output:**

* read1.fastq\_filtered\_trimmed (here)
* read2.fastq\_filtered\_trimmed (here)

**Parameters Description:**

* -i <Forward read/sequence file>
* -irev <Reverse read/sequence file of paired-end data>
* -q | -qualCutOff <Integer> (Only for FASTQ files) Cut-off PHRED quality score for trimming reads from right end (3' end)

**Reference:**

Patel et al. (2012) “NGS QC Toolkit: A Toolkit for Quality Control of Next Generation Sequencing Data. *PloS ONE*

**Note: Need to check \*\_stat file from Step 1 to see how much data is filtered if filter above certain threshold then we need to take decision whether to proceed for further analysis or not.**

**Step2: Alignment**

**Module Requirements:**

* module load samtools/0.1.18
* module load bwa/0.5.10

**2.1 Running BWA “aln” step**

**Usage:**

bwa aln -t 12 /pathTo/hg19.fasta <read1.fastq\_filtered\_trimmed> > aln1.sai

bwa aln -t 12 /pathTo/hg19.fasta <read2.fastq\_filtered\_trimmed> > aln2.sai

**Input:**

* Reference Genome Indexed with BWA [required]
* Read file forward; passed QC [required]
* Read file reverse; passed QC [required]

Note: Need to run bwa “aln” separately for forward and reverse file

**Output:**

* Suffix array index file; **provide file names** (aln1.sai and aln2.sai here)

**Parameters Description:**

|  |  |
| --- | --- |
| * **-t** *INT* | Number of threads (multi-threading mode) |

**2.2 Running BWA “sampe” step**

**Usage:**

bwa sampe –r @RG"\t"ID:<SAMPLE1\_RG1>"\t"LB:<SAMPLE1\_LIB1>"\t"SM:<SAMPLE1>"\t"PL:<ILLUMINA>/pathTo/reference.fasta <aln1.sai> <aln2.sai> <read1.fastq\_filtered\_trimmed> <read2.fastq\_filtered\_trimmed> > aln.sam

**Input:**

* Suffix array index file (aln1.sai and aln2.sai here) [required]
* Input fasta file (read1.fastq\_filtered\_trimmed read2.fastq\_filtered\_trimmed here)[required]
* -r: read group information (required for downstream analysis)

**Output:**

* \*.sam Alignment file in SAM format; **provide file names** (aln.sam here)

**Parameters Description:**

|  |  |
| --- | --- |
| * **-r STR** | Specify the read group in a format like ‘@RG\tID:foo\tSM:bar’. [null] |

Note: If we continue to use the same conventions as GES uses for Genologics and the HiSeq at JAX BHB, the read group info can be derived from the fastq file.

Given a pair of fastq files, the @RG info can be pulled from the name (of either of pair) and the first line of the file. Highlighted colors match read group info needed.

0916-6180\_GES12\_36721\_TTAGGC\_L007\_R2\_ALL.fastq

@HISEQ2000:158:D0R99ACXX:7:1101:1387:1965 2:N:0:TTAGGC

GTGTCTGTGGGGCCTTCCGCTCTGGCTCCTGTGGA

+

BB@FFFFFHHHHGJJFIJJIGIJJJIGJJJEBGII

@RG

ID HISEQ2000:158:D0R99ACXX:7 (the machine name, run#, flowcell ID, lane #)

LB GES12\_36721\_TTAGGC (GES sample ID with barcode)

PL ILLUMINA (platform)

SM 0916-6180 (customer sample ID)

**References:**

Li et al. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*

Li et al. (2009). "The Sequence Alignment/Map format and SAMtools." *Bioinformatics*

**Step3: Variant Pre-processing**

**Module requirements (or path):**

* module load python
* module load java
* module load R
* /opt/compsci/picard/1.8.4/
* /opt/compsci/GATK/2.2-16/GenomeAnalysisTK.jar

**Step3.1: Picard SortSam.jar (generating sorted alignment bam file)**

**Usage:**

java -Xmx2g -jar picard/SortSam.jar \

SO=coordinate \

INPUT=<aln.sam> \

OUTPUT=aln.bam \

VALIDATION\_STRINGENCY=LENIENT \

CREATE\_INDEX=true

**Input:**

* The BWA generated sam file (aln.sam here) [required]

**Output:**

* The coordinate sorted alignment file in bam format; **provide file name** (aln.bam here)
* Index of coordinate sorted alignment file (aln.bai here)

**Parameters Description:**

|  |  |
| --- | --- |
| * SO: SORT\_ORDER. If not supplied OUTPUT is in the same order as INPUT. Default value: null. Possible values: {unsorted, queryname, coordinate} |  |

* VALIDATION\_STRINGENCY: Validation stringency for all SAM files read by this program. Setting stringency to SILENT can improve performance when processing a BAM file in which variable-length data (read, qualities, tags) do not otherwise need to be decoded. Default value: STRICT. This option can be set to 'null' to clear the default value. Possible values: {STRICT, LENIENT, SILENT}
* CREATE\_INDEX: Whether to create a BAM index when writing a coordinate-sorted BAM file. Default value: false. This option can be set to 'null' to clear the default value. Possible values: {true, false}

**Step3.2: Picard Mark Duplicates (Removing duplicates in BAM file)**

**Usage:**

java -Xmx2g -jar picard/MarkDuplicates.jar \

INPUT=<aln.bam>  \

OUTPUT=aln\_dedup.bam \

METRICS\_FILE=metrics \

REMOVE\_DUPLICATES=true \

CREATE\_INDEX=true \

VALIDATION\_STRINGENCY=LENIENT

**Input:**

* Coordinated sorted BAM file (aln.bam here) [required]
* File to write duplication metrics to (metrics here) [Required]

**Output:**

* Duplicate removed bam file; **provide file name** (aln\_dedup.bam here)
* Index of duplicate removed bam file (aln\_dedup.bai here)

**Parameters Description:**

* REMOVE\_DUPLICATES: If true do not write duplicates to the output file instead of writing them with appropriate flags set. Default value: false. This option can be set to 'null' to clear the default value. Possible values: {true, false}
* METRICS\_FILE: File to write duplication metrics

**Step3.3: Realignment around indels Part I (Target Interval Creation)**

**Usage:**

java -Xmx2g -jar GenomeAnalysisTK.jar \

-I <aln\_dedup.bam> \

-R hg19.fasta \

--num\_threads 8 \

-T RealignerTargetCreator \

-o forIndelRealigner.intervals \

[-known /pathTo/<indels.vcf>]

**Input**

* Duplicated removed bam file (aln\_dedup.bam here) [required]
* Reference genome file (hg19.fasta here) [required]

**Output**

* Interval file output to be used by IndelRealigner; **provide file name** (forIndelRealigner.intervals here)

**Parameters Description:**

* -known: Input VCF file with known indels
* --num\_threads: How many data threads should be allocated to running this analysis

**Databases could be used with this command (-known):**

* Mills\_and\_1000G\_gold\_standard.indels.hg19.vcf
* 1000G\_phase1.indels.hg19.vcf (currently from the 1000 Genomes Phase I indel calls)

Note: index files (\*.idx) must be available in same directory as the \*.vcf files

Note: we will use “known” files provided by the Broad GATK bundle. We will use the files that best correspond to the reference genome used.

**Step3.4: Realignment around indels Part II (Performing the local re-alignment)**

**Usage:**

java -Xmx4g -jar GenomeAnalysisTK.jar \

-I <aln\_dedup.bam> \

-R hg19.fasta \

-T IndelRealigner \

-targetIntervals <forIndelRealigner.intervals> \

-o realignedBam.bam \

[-known /pathTo/<indels.vcf>] \

**Input:**

* Duplicated removed bam file (aln\_dedup.bam here) [required]
* Reference genome file (hg19.fasta here) [required]
* Interval file created by RealignerTargetCreator (forIndelRealigner.intervals here) [required]

**Output:**

* Bam file with realigned indels; **provide file name** (realignedBam.bam here)
* Index file of realigned bam file (realignedBam.bai here)

**Parameters Description:**

* -known: Input VCF file with known indels

**Databases could be used with this command (-known):**

* Mills\_and\_1000G\_gold\_standard.indels.hg19.vcf
* 1000G\_phase1.indels.hg19.vcf (currently from the 1000 Genomes Phase I indel calls)

**Step3.5: Base Quality Recalibration Part I**

Notes: Requires R packages (see appendix) to create graphs.

**Usage:**

java -Xmx4g -jar GenomeAnalysisTK.jar \

-T BaseRecalibrator \

-I <realignedBam.bam> \

- nct 8

-R resources/hg19.fasta \

-knownSites bundle/hg19/<dbsnp\_137.hg19.vcf> \

-knownSites another/optional/<setOfSitesToMask.vcf> \

-o <recal\_data.grp

--plot\_pdf\_file Pre\_recal\_data.pdf

**Input:**

* Realigned file realignedBam.bam (here) [required]
* Reference genome file (hg19.fasta here) [required]
* -knownSites (database of known SNP and Indel sites) [not required]
* File name to store pre recalibration plots (Pre\_recal\_data.pdf here) [not required]

**Output:**

* output recalibration table file Recal\_data.grp
* recalibration plots pdf file

**Parameters Description:**

* -knownSites: A database of known polymorphic sites to skip over in the recalibration algorithm
* --plot\_pdf\_file: The output recalibration pdf file to create

**Databases could be used with this command (-knownSites):**

* The most recent dbSNP release (build ID > 132) [dbsnp\_137.hg19.vcf]
* Mills\_and\_1000G\_gold\_standard.indels.hg19.vcf
* 1000G\_phase1.indels.hg19.vcf (currently from the 1000 Genomes Phase I indel calls)

**Step 3.5: Base Quality Recalibration Part II (**Just creating plots after model generation)

Usage:

java -Xmx4g -jar GenomeAnalysisTK.jar \

-T BaseRecalibrator \

-I <realignedBam.bam> \

-nct 8 \

-R resources/hg19.fasta \

-BQSR recal\_data.grp \

-knownSites bundle/hg18/</dbsnp\_132.hg18.vcf> \

-knownSites another/optional/</setOfSitesToMask.vcf> \

-o Post\_recal\_data.grp

--plot\_pdf\_file Post\_recal\_data.pdf

**Input:**

* Realigned file realignedBam.bam (here) [required]
* Reference genome file (hg19.fasta here) [required]
* output recalibration table file recal\_data.grp [required]
* -knownSites (database of known SNP and Indel sites) [not required unless supplied in previous step]
* File name to store pre recalibration plots (Post\_recal\_data.pdf here) [not required]

**Output:**

* post recalibration table file
* Post recalibration plots pdf file

**Parameters Description:**

* -knownSites: A database of known polymorphic sites to skip over in the recalibration algorithm
* --plot\_pdf\_file: The output recalibration pdf file to create
* -BQSR: The input covariates table file which enables on-the-fly base quality score recalibration

**Databases could be used with this command (-knownSites):**

* The most recent dbSNP release (build ID > 132) [dbsnp\_137.hg19.vcf]
* Mills\_and\_1000G\_gold\_standard.indels.hg19.vcf
* 1000G\_phase1.indels.hg19.vcf (currently from the 1000 Genomes Phase I indel calls)

**Step3.5: Base Quality Recalibration Part III (**printing the reads**)**

**Usage:**

java -Xmx4g -jar GenomeAnalysisTK.jar \

-T PrintReads \

-R hg19.fasta \

-nct 8

-I <realignedBam.bam> \

-BQSR recal\_data.grp \

-o realigned\_BQSR.bam

**Input:**

* Realigned file realignedBam.bam (here) [required]
* Reference genome file (hg19.fasta here) [required]
* output recalibration table file recal\_data.grp [required]
* Outfile name with fixed base qualities [required]

**Output:**

* Bamfile with fixed base qualities (realigned\_BQSR.bam here)
* Index files of Bam alignment (realigned\_BQSR.bai here)

**Parameters Description:**

* PrintReads (commands to print read from fixed BQSR file)

Note: The file size becomes almost double after BQSR step as additional information been added to sam fields.

**Step 3.6: Picard CalculateHsMetrics (**generating target enrichment information**)**

**Usage:**

java -Xmx2g -jar CalculateHsMetrics.jar \

BAIT\_INTERVALS=<bait.txt> \

TARGET\_INTERVALS=<target.txt> \

INPUT=<realigned\_BQSR.bam> \

OUTPUT=Metricsfile.txt \

VALIDATION\_STRINGENCY=LENIENT \

**Input**

* Realigned base quality fixed file realigned\_BQSR.bam (here) [required]
* bait interval file (bait here) [required]
* target interval file (target here) [required]

**Output:**

* Output file with target enrichment information (Metricsfile.txt here) [required]

**Parameters Description:**

* BAIT\_INTERVAL: An interval list file that contains the locations of the baits used. [Required]
* TARGET\_INTERVALS: An interval list file that contains the locations of the targets [Required]

**References:**

* **McKenna A et al. (2010) The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data Genome Res**
* **DePristo et al. (2011) A framework for variation discovery and genotyping using next-generation DNA sequencing data. Nature Genetics**
* <http://picard.sourceforge.net>

**Step 4: Variant Calling**

**Module requirements:**

* module load python
* module load java
* module load R
* /opt/compsci/GATK/2.2-16/GenomeAnalysisTK.jar

**Usage:**

java -Xmx2g -jar GenomeAnalysisTK.jar \

-R resources/hg19.fasta \

-T UnifiedGenotyper \

-nt 8

-I <realigned\_BQSR.bam> \

-glm BOTH \

--dbsnp <dbSNP.vcf> \

-o variants.raw.vcf \

-stand\_call\_conf [50.0] \

-stand\_emit\_conf 10.0 \

-dcov [50 for 4x, 200 for >30x WGS or Whole exome] \

[-L <targets.interval\_list>]

**Input:**

* Realigned base quality fixed file realigned.fixed\_BQSR.bam (here) [required]
* Reference genome file (hg19.fasta here) [required]

**Output:**

* SNP and indel calls in variant call format (snps.raw.vcf here)
* VCF index file (snps.raw.vcf.idx here)

**Parameter description:**

-stand\_call\_conf: The minimum phred-scaled confidence threshold at which variants should be called.

-stand\_emit\_conf: The minimum phred-scaled confidence threshold at which variants should be emitted (and filtered with LowQual if less than the calling threshold). This argument allows you to emit low quality calls as filtered records.

**Databases could be used with this command:**

These tools do NOT require known sites, but if SNPs are provided with the -dbsnp argument they will use them for variant annotation. We use this file:

* The most recent dbSNP release (build ID > 132) [dbsnp\_137.hg19.vcf]
* Bed file for target intervals -L target\_intervals.bed (agilent target list)

Note: Bed file must match reference genome used for alignment (eg, it should not contain chr\_un or chr\_random if using UCSC hg19).

**Step5: Variant Quality Recalibration**

Note: The UnifiedGenotyper uses a fundamentally different likelihood model when calling different classes of variation and so therefore the VQSR must be run separately for SNPs and INDELs to build separate adaptive error models.

**Module requirements:**

* module load python
* module load java
* module load R
* /opt/compsci/GATK/2.2-16/GenomeAnalysisTK.jar

**Step 5.1: Generating SNP model**

**Usage:**

java -Xmx4g -jar GenomeAnalysisTK.jar \

-T VariantRecalibrator \

-R reference/hg19.fasta \

--num\_threads 8 \

-input <variants.raw.vcf> \

-resource:<hapmap,known=false,training=true,truth=true,prior=15.0 hapmap\_3.3.hg19.vcf> \

-resource:<omni,known=false,training=true,truth=false,prior=12.0 1000G\_omni2.5.hg19.vcf> \

-resource:<dbsnp,known=true,training=false,truth=false,prior=6.0 dbsnp\_137.hg19.vcf> \

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

-mode SNP \

-recalFile pathTo/output.recal \

-tranchesFile pathTo/output.tranches \

-rscriptFile pathTo/output.plots.R \

--maxGaussians 4 \

-percentBad 0.05

**Input:**

* Reference genome file (hg19.fasta here) [required]
* SNP/indels calls in variant call format (variants.raw.vcf here) [required]

**Output:**

* Recal file to be used by ApplyRecalibration; **provide name** (output.recal)
* Tranches file to be used by ApplyRecalibration; **provide name** (output.tranches)
* The output rscript file generated by the VQSR to aid in visualization of the input data and learned model

**Parameter description:**

* -mode: Recalibration mode to employ: 1.) SNP for recalibrating only snps (emitting indels untouched in the output VCF); 2.) INDEL for indels
* -rscriptFile: The output rscript file generated by the VQSR to aid in visualization of the input data and learned model
* **-**resource: A list of sites for which to apply a prior probability of being correct but which aren't used by the algorithm
* -an: The names of the annotations which should used for calculations. Description given below:

1. **QD: QualByDepth**: Variant confidence (from the QUAL field) / unfiltered depth Low scores are indicative of false positive calls and artifacts. Note that QualByDepth requires sequencing reads associated with the samples with polymorphic genotypes.
2. **HaplotypeScore:** Consistency of the site with two (and only two) segregating haplotypes. Higher scores are indicative of regions with bad alignments, often leading to artifactual SNP and indel calls. Note that the Haplotype Score is only calculated for sites with read coverage.
3. **MQRankSum:** The u-based z-approximation from the Mann-Whitney Rank Sum Test for mapping qualities (reads with ref bases vs. those with the alternate allele) Note that the mapping quality rank sum test cannot be calculated for sites without a mixture of reads showing both the reference and alternate alleles. )
4. **ReadPosRankSum:** The u-based z-approximation from the Mann-Whitney Rank Sum Test for the distance from the end of the read for reads with the alternate allele; if the alternate allele is only seen near the ends of reads this is indicative of error). Note that the read position rank sum test cannot be calculated for sites without a mixture of reads showing both the reference and alternate alleles.
5. **FS:** Phred-scaled p-value using Fisher's Exact Test to detect strand bias (the variation being seen on only the forward or only the reverse strand) in the reads? More bias is indicative of false positive calls. Note that the fisher strand test may not be calculated for certain complex indel cases or for multi-allelic site
6. **MQ: Mapping Quality**
7. **InbreedingCoeff:** Likelihood-based (using PL field) test for the inbreeding among samples. A continuous generalization of the Hardy-Weinberg test for disequilibrium that works well with limited coverage per sample. See the 1000 Genomes Phase I release for more information. Note that the Inbreeding Coefficient will not be calculated for files with fewer than a minimum (generally 10) number of samples.

**Databases could be used with this command:**

* -resource:hapmap,VCF,known=false,training=true,truth=true,prior=15.0 hapmap\_3.3. hg19.vcf \
* -resource:omni,VCF,known=false,training=true,truth=false,prior=12.0 1000G\_omni2.5. hg19.vcf \
* -resource:dbsnp,VCF,known=true,training=false,truth=false,prior=6.0 dbsnp\_137. hg19.vcf \

**Step 5.2: Generating Indel model**

**Usage:**

java -Xmx4g -jar GenomeAnalysisTK.jar \

-T VariantRecalibrator \

-R reference/hg19.fasta \

--num\_threads 8 \

-input <variants.raw.vcf> \

-resource:mills,known=true,training=true,truth=true,prior=12.0 pathTo/Mills\_and\_1000G\_gold\_standard.indels.hg19.vcf \

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

-mode INDEL \

--maxGaussians 4 \

-minNumBadVariants 500 \

-recalFile pathTo/output\_indel.recal \

-tranchesFile pathTo/output\_indel.tranches \

-rscriptFile pathTo/output\_indel.plots.R

**Input:**

* Reference genome file (hg19.fasta here) [required]
* SNP/indels calls in variant call format (variants.raw.vcf here) [required]

**Output:**

* Recal file to be used by ApplyRecalibration; **provide name** (output\_indel.recal)
* Tranches file to be used by ApplyRecalibration; **provide name** (output\_indel.tranches)
* The output rscript file generated by the VQSR to aid in visualization of the input data and learned model

**Parameter description:**

* No new parameter except analysis mode changed to INDEL

**Databases could be used with this command:**

-resource:mills,VCF,known=false,training=true,truth=true,prior=12.0 [Mills\_and\_1000G\_gold\_standard.indels.hg19.]

**Step 5.3: Apply SNP recalibration**

**Usage:**

java -Xmx3g -jar GenomeAnalysisTK.jar \

-T ApplyRecalibration \

-R reference/hg19.fasta \

--num\_threads 8 \

-input <variants.raw.vcf> \

--ts\_filter\_level 99.0 \

-tranchesFile pathTo/output.tranches \

-recalFile pathTo/output.recal \

-mode SNP \

-o output.recalibrated.filtered.vcf

**Input:**

* Reference genome file (hg19.fasta here) [required]
* SNP/indels calls in variant call format (variants.raw.vcf here) [required]
* Recal file generated by VariantRecalibrator (output.recal) [required]
* tranches file generated by VariaintRecalibrator (output.tranches) [required]

**Output:**

* Output file with recalibrated SNPs; **provide name** (output.recalibrated.filtered.vcf here)
* Index vcf file (output.recalibrated.filtered.vcf.idx)

**Parameter description:**

* --ts\_filter\_level: The truth sensitivity level at which to start filtering
* -mode: Recalibration mode to employ: 1.) SNP for recalibrating only SNPs (emitting indels untouched in the output VCF); 2.) INDEL for indels

**Step 5.4: Apply Indel recalibration**

Note: run after SNP recalibration finished

**Usage:**

java -Xmx3g -jar GenomeAnalysisTK.jar \

-T ApplyRecalibration \

-R reference/hg19.fasta \

--num\_threads 8 \

-input <output.recalibrated.filtered.vcf> \

--ts\_filter\_level 95.0 \

-tranchesFile pathTo/output\_indel.tranches \

-recalFile pathTo/output\_indel.recal \

-mode INDEL \

-o analysis\_ready.vcf

**Input:**

* Reference genome file (hg19.fasta here) [required]
* SNP calls in variant call format (snps.raw.vcf here) [required]
* Recal file generated by VariantRecalibrator (output\_indel.recal) [required]
* tranches file generated by VariaintRecalibrator (output\_indel.tranches) [required]

**Output:**

* Output file with analysis ready variants; **provide name** (analysis\_ready.vcf here)
* Output variant index file (analysis\_ready.vcf.idx here)

**Parameter description:**

* No new parameter here except mode is INDEL

**Broad Recommended parallelism configurations (Guide Article #1975 2013-01-14 18:02:57)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tool** | **Full name** | **Type of traversal** | **NT** | **NCT** | **SG** |
| RTC | RealignerTargetCreator | RodWalker | + | - | - |
| IR | IndelRealigner | ReadWalker | - | - | + |
| BR | BaseRecalibrator | LocusWalker | - | + | + |
| PR | PrintReads | ReadWalker | - | + | - |
| RR | ReduceReads | ReadWalker | - | - | + |
| UG | UnifiedGenotyper | LocusWalker | + | + | + |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tool** | **RTC** | **IR** | **BR** | **PR** | **RR** | **UG** |
| Available modes | NT | SG | NCT,SG | NCT | SG | NT,NCT,SG |
| Cluster nodes | 1 | 4 | 4 | 1 | 4 | 4 / 4 / 4 |
| CPU threads (-nct) | 1 | 1 | 8 | 4-8 | 1 | 3 / 6 / 24 |
| Data threads (-nt) | 24 | 1 | 1 | 1 | 1 | 8 / 4 / 1 |
| Memory (Gb) | 48 | 4 | 4 | 4 | 4 | 32 / 16 / 4 |

**Broad recommended sets of known sites per tool** (http://gatkforums.broadinstitute.org/discussion/1247/what-should-i-use-as-known-variantssites-for-running-tool-x)

**Summary table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tool | dbSNP 129 | dbSNP >132 | Mills indels | 1KG indels | HapMap | Omni |
| RealignerTargetCreator |  |  | X | X |  |  |
| IndelRealigner |  |  | X | X |  |  |
| BaseRecalibrator |  | X | X | X |  |  |
| (UnifiedGenotyper/ HaplotypeCaller) |  | X |  |  |  |  |
| VariantRecalibrator |  | X | X |  | X | X |
| VariantEval | X |  |  |  |  |  |

**RealignerTargetCreator and IndelRealigner**

These tools require known indels passed with the -known argument to function properly. We use both the following files:

* Mills\_and\_1000G\_gold\_standard.indels.b37.sites.vcf
* 1000G\_phase1.indels.b37.vcf (currently from the 1000 Genomes Phase I indel calls)

**BaseRecalibrator**

This tool requires known SNPs and indels passed with the -knownSites argument to function properly. We use all the following files:

* The most recent dbSNP release (build ID > 132)
* Mills\_and\_1000G\_gold\_standard.indels.b37.sites.vcf
* 1000G\_phase1.indels.b37.vcf (currently from the 1000 Genomes Phase I indel calls)

**UnifiedGenotyper / HaplotypeCaller**

These tools do NOT require known sites, but if SNPs are provided with the -dbsnp argument they will use them for variant annotation. We use this file:

* The most recent dbSNP release (build ID > 132)

**VariantRecalibrator**

This tool requires known SNPs and indels passed with the -resource argument to function properly. We use all the following files:

* HapMap genotypes and sites
* OMNI 2.5 genotypes and sites for 1000 Genomes samples
* The most recent dbSNP release (build ID > 132)
* Mills\_and\_1000G\_gold\_standard.indels.b37.sites.vcf

For best results, these resources should be passed with these parameters:

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

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

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

-resource:mills,VCF,known=false,training=true,truth=true,prior=12.0 gold.standard.indel.b37.vcf

**Appendix**

**Files required**

Human reference genome

Illumina adapter sequences used

Desired ‘known’ files from Broad GATK bundle

Interval files for exome capture bait and target (create in house from vendor’s files)

Bedfile for Unified genotyper (bait/target from vendor)

**Accessory applications required**

R packages:

library("ggplot2")

library(gplots)

library("reshape")

library("grid")

library("tools") #For compactPDF in R 2.13+

library(gsalib)

(check R version compatibilities)

**Decision points-open questions**

Need to check \*\_stat file from Step 1 to see how much data is filtered if filter above certain threshold then we need to take decision whether to proceed for further analysis or not.

Need to determine/validate VQSR with respect to tumor samples. Parameter settings to be determined (if VQSR) is used. For example, INDEL model run on single exome sample failed unless minNumBadVariants set to 500 from default 2500, but this is considered dangerous.

Will we use additional variant caller(s)? Possible candidates, SNVmix2, Varscan2.