全基因组测序分析：

GATK4 call germline SNPs and Indels

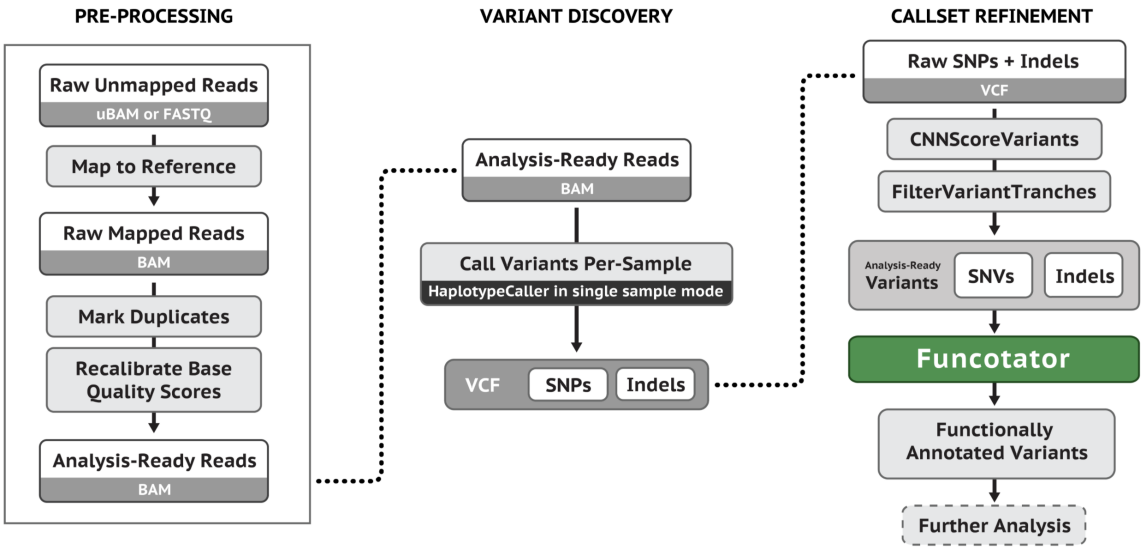
> **参考：**
>
> GATK4流程学习之DNA-Seq variant calling(Germline：SNP+INDEL)
>
> GATK：Germline short variant discovery (SNPs + Indels)
>
> 如何正确设置GATK VQSR的模型训练参数

全基因组测序分析：
GATK4 call germline SNPs and Indels
1. 前期准备
   1.1 工作目录
   1.2 数据下载
   1.3 软件安装
      1.3.1 GATK软件安装
      1.3.2 conda安装软件
2. 测序数据质控
   2.1 质量评估：fastqc、multiqc
   2.2 质控和去接头：trim_galore
3. 比对到参考基因组
   3.1 对参考基因组建立索引
   3.2 比对到参考基因组
4. GATK 流程
   4.1 标记或去除重复序列 MarkDuplicates
   4.2 碱基质量重校正BQSR
   4.3 变异检测
      4.3.1 HaplotypeCaller call variant
      4.3.2 Consolidate GVCFs [可选：多样本合并]
   4.4 变异过滤
      4.4.1 SNP 过滤
      4.4.2 Indels 过滤
5. 变异注释

汇总： 多样本批量处理脚本
0. 注意
1. 质控：fastqc.sh、trim_galore.sh
2. 比对：bwa.sh
3. GATK：标记去重 MarkDuplicates.sh
4. GATK：碱基质量重校正 bqsr.sh
5. GATK：变异检测 HaplotypeCaller.sh
6. GATK：多样本合并 mergevcf.sh
7. GATK：变异过滤 VQSR.sh

GATK官网提供了call germline short variants的best practice

单样本 *Main steps for Germline Single-Sample Data:*



We begin by calling variants per sample in order to produce a file in GVCF format. Next, we consolidate GVCFs from multiple samples into a GenomicsDB datastore. We then perform joint genotyping, and finally, apply VQSR filtering to produce the final multisample callset with the desired balance of precision and sensitivity.

多样本 *Main steps for Germline Cohort Data:*

Single sample variant discovery uses HaplotypeCaller in its default single-sample mode to call variants in an analysis-ready BAM file. The VCF that HaplotypeCaller emits errs on the side of sensitivity, so some filtering is often desired. To filter variants first run the CNNScoreVariants tool. This tool annotates each variant with a score indicating the model's prediction of the quality of each variant. To apply filters based on those scores run the FIlterVariantTranches tool with SNP and INDEL sensitivity tranches appropriate for your task.

1. 前期准备

1.1 工作目录

```
## 在用户的根目录下创建 wgs、biosoft、biodata
# 以wgs作为工作目录，biosoft目录存放相关软件，biodata目录存放数据库文件
mkdir wgs biosoft biodata
cd wgs

## 在 ~/wgs 中创建 bin、input、output 三个文件夹
mkdir -p bin input output/{qc,align,gatk}

# bin：存放所有执行程序和代码
# input：存储所有输入数据
# output：存储输出数据
```

1.2 数据下载

参考基因组：hg19

```
cd ~/biodata

## UCSC hg19
wget -c https://hgdownload.soe.ucsc.edu/goldenPath/hg19/bigZips/hg19.fa.gz

## GATK bundle 下载dbSNP、1000G相关数据(包括VCF文件和对应的idx文件)，用于GATK best practice过程中变异的校正
gatk_FTP=ftp://gsapubftp-anonymous@ftp.broadinstitute.org/bundle/hg19
wget -c $gatk_FTP/dbsnp_138.hg19.vcf.gz
wget -c $gatk_FTP/dbsnp_138.hg19.vcf.idx.gz
wget -c $gatk_FTP/hapmap_3.3.hg19.sites.vcf.gz
wget -c $gatk_FTP/hapmap_3.3.hg19.sites.vcf.idx.gz
wget -c $gatk_FTP/1000G_omni2.5.hg19.sites.vcf.gz
wget -c $gatk_FTP/1000G_omni2.5.hg19.sites.vcf.idx.gz
wget -c $gatk_FTP/1000G_phase1.indels.hg19.sites.vcf.gz
wget -c $gatk_FTP/1000G_phase1.indels.hg19.sites.vcf.idx.gz
wget -c $gatk_FTP/1000G_phase1.snps.high_confidence.hg19.sites.vcf.gz
wget -c $gatk_FTP/1000G_phase1.snps.high_confidence.hg19.sites.vcf.idx.gz
wget -c $gatk_FTP/Mills_and_1000G_gold_standard.indels.hg19.sites.vcf.gz
wget -c $gatk_FTP/Mills_and_1000G_gold_standard.indels.hg19.sites.vcf.idx.gz

nohup wget -c ftp://gsapubftp-anonymous@ftp.broadinstitute.org/bundle/hg19/1000G_phase1.snps.high_confidence.hg19.sites.vcf.gz &
nohup wget -c ftp://gsapubftp-anonymous@ftp.broadinstitute.org/bundle/hg19/1000G_phase1.snps.high_confidence.hg19.sites.vcf.idx.gz &

## 解压上述文件（hg19版本必须，不然后续报错）
find . -name "*.gz" |xargs gzip -d # 批量解压，不保留原文件
```

1.3 软件安装

1.3.1 GATK软件安装

```
## GATK官网获取下载链接
cd biosoft
wget -c https://github.com/broadinstitute/gatk/releases/download/4.2.0.0/gatk-4.2.0.0.zip
unzip gatk-4.2.0.0.zip

vi ~/.bashrc

## 添加到环境，将以下内容添加到 .bashrc文件中
------------------------------------------------------------------------------
# added by gatk4
export PATH="/home/kongyongqiang/biosoft/gatk4/gatk-4.2.0.0:$PATH"
------------------------------------------------------------------------------
```

```
source ~/.bashrc
```

1.3.2 conda安装软件

```
## 新建环境 wgs
conda create -n wgs python=3

## 激活 wgs 环境
conda activate wgs    # source activate wgs

## 安装必要的软件
# sra-tools fastqc trim-galore multiqc bwa samtools qualimap vcftools bedtools
cnvkit 等
# 注:multiqc 依赖 python2.7+,3.4+或3.5+
conda install -y sra-tools fastqc trim-galore multiqc bwa samtools qualimap
vcftools bedtools cnvkit
```

2. 测序数据质控

2.1 质量评估：fastqc、multiqc

```
## 原始下机数据
fq1 = ~/wgs/input/sample_1.fastq.gz
fq2 = ~/wgs/input/sample_2.fastq.gz

cd ~/wgs/out/qc
## fastqc 质量评估
nohup fastqc -o ./ -t 16 $fq1 >> ./sample_1_fastqc.log 2>&1 &
nohup fastqc -o ./ -t 16 $fq2 >> ./sample_2_fastqc.log 2>&1 &

## multiqc 整合结果，方便批量查看
multiqc ./*zip -o ./multiqc
```

2.2 质控和去接头：trim_galore

去除低质量碱基，过滤低质量reads，去除接头

```
cd ~/wgs/output/qc

## trim_galore 质控去接头
nohup trim_galore \
    --paired -q 28 \
    --phred33 \
    --length 30 \
    --stringency 3 \
    --gzip \
    --cores 8 \
    -o ./ $fq1 $fq2 >> ./sample_trim.log 2>&1\
    && echo "** trim_galore done **"

## 再次使用 fastqc 进行质控评估
```

3. 比对到参考基因组

质控后的 fastq文件，需要比对到参考基因组上才能让这些数据有意义

3.1 对参考基因组建立索引

在比对之前先对参考基因组建立索引

```
cd ~/biodata
conda activate wgs

gunzip -c hg19.fa.gz > hg19.fa

# samtools 构建索引，生成 hg19.fa.fai文件
samtools faidx hg19.fa

# bwa index 构建参考基因组索引
time bwa index -a bwtsw hg19.fa

#  构建dict文件
time gatk --java-options "-Xmx20G -Djava.io.tmpdir=./tmp"
CreateSequenceDictionary \
    -R hg19.fa \
    -O hg19.fa.dict \
    && echo "** dict done **"
```

3.2 比对到参考基因组

```
cd ~/wgs/output/align

## 质控后的fastq文件和参考基因组index
```

```
fq1= ~/wgs/input/sample_1_val_1.fq.gz
fq2= ~/wgs/input/sample_2_val_2.fq.gz
ref= ~/biodata/hg19.fa

## bwa mem比对、samtools sort排序，生成bam文件
time bwa mem -M -t 16 \
    -R '@RG\tID:sample\tPL:illumina\tLB:WGS\tSM:sample' \
    $ref $fq1 $fq2 \
    |samtools sort -@ 10 -m 4G -o ./sample.bam \
    && echo "** bwa mapping done **"
    # -R 设置reads标头， “\t”分割
```

4. GATK 流程

4.1 标记或去除重复序列 MarkDuplicates

拿到比对后的bam文件之后，要进行标记重复序列甚至去除掉重复序列。用到了**GATK**的**MarkDuplicates**工具，代码如下：

```
cd ~/wgs/output/gatk
mkdir tmp  # 存放运行中的tmp文件

## GATK MarkDuplicates 标记或去除重复序列
gatk --java-options "-Xmx20G -Djava.io.tmpdir=./tmp" MarkDuplicates \
    -I ./sample.bam \
    --REMOVE_DUPLICATES true \
    -O ./sample_marked.bam \
    -M ./sample.metrics \
    1>./sample_log.mark 2>&1 \
    && echo "** MarkDuplicates done **"

## samtools 对生成的bam文件构建索引,这是后续步骤所需要的
samtools index -@ 16 -m 4G -b ./sample_marked.bam ./sample_marked.bai
```

4.2 碱基质量重校正BQSR

碱基的质量值由测序仪和测序系统产生，机器带来的系统误差和噪声是不可避免的。因此，找变异之前，需要进行碱基质量重校正**BQSR** （Base Quality Score Recalibration）。用到了**GATK**的两个工具**BaseRecalibrator**和**ApplyBQSR**，代码如下：

```
cd ~/wgs/output/gatk

## 数据库文件和参考基因组
snp= ~/biodata/dbsnp_138.hg19.vcf
indel= ~/biodata/1000G_phase1.indels.hg19.sites.vcf
mills= ~/biodata/Mills_and_1000G_gold_standard.indels.hg19.sites.vcf
ref= ~/biodata/hg19.fa

## GATK BaseRecalibrator 计算所有需重校正的reads和特征值,输出为一份校准表文件
time gatk --java-options "-Xmx20G -Djava.io.tmpdir=./tmp"  BaseRecalibrator \
```

```
    -R $ref  \
    -I ./sample_marked.bam \
    --known-sites ${snp} \
    --known-sites ${indel} \
    --known-sites ${mills} \
    -O ./sample_recal.table \
    1>./sample_log.recal 2>&1 \
    && echo "** BaseRecalibrator done **"

## GATK ApplyBQSR 利用校准表文件,重新调整原来BAM文件中的碱基质量值，并使用新的质量值重新输出一份新的BAM文件
time gatk --java-options "-Xmx20G -Djava.io.tmpdir=./tmp"  ApplyBQSR \
    -R $ref \
    -I ./sample_marked.bam \
    -bqsr ./sample_recal.table \
    -O ./sample_bqsr.bam \
    1>./sample_log.ApplyBQSR  2>&1
    && echo "** ApplyBQSR done **"


## samtools 对生成的sample_bqsr.bam文件构建索引,这是后续步骤所需要的
time samtools index ./sample_bqsr.bam
```

4.3 变异检测

4.3.1 HaplotypeCaller call variant

使用HaplotypeCaller对上述的bam文件进行call variant，生成一个VCF文件用于后续位点的质控和注释

```
dbsnp= ~/biodata/dbsnp_138.hg19.vcf
bam= ~/wgs/output/gatk/sample_bqsr.bam
ref= ~/biodata/hg19.fa

# GATK HaplotypeCaller
time gatk --java-options "-Xmx20G -Djava.io.tmpdir=./tmp" HaplotypeCaller \
    -R $ref \
    -I $bam \
    --dbsnp $dbsnp \
    -O ./sample_HC.vcf \
    1>./sample_log.HC 2>&1 \
    && echo "** HaplotypeCaller done **"
```

4.3.2 Consolidate GVCFs [可选：多样本合并]

```
ref = ~/biodata/hg19.fa

## GATK GenomicsDBImport
for chr in chr{1..22} chrX chrY chrM
do
    time gatk --java-options "-Xmx20G -Djava.io.tmpdir=./" GenomicsDBImport \
        -R $ref \
        $(ls ./*HC.vcf | awk '{print "-V "$0" "}') \
        -L $chr \
        --genomicsdb-workspace-path gvcfs_$chr.db \
```

```
        1>./sample_log.GenomicsDBImport 2>&1

    ## GATK GenotypeGVCFs
    time gatk --java-options "-Xmx20G -Djava.io.tmpdir=./tmp" GenotypeGVCFs \
        -R $ref \
        -V gendb://gvcfs_$chr.db \
        -O ./gvcfs_$chr.vcf \
        1>./sample_log.GenotypeGVCFs 2>&1
done

## GATK GatherVcfs 合并所有染色体的vcf文件
time gatk --java-options "-Xmx20G -Djava.io.tmpdir=./tmp" GatherVcfs \
    $(for i in {1..22} X Y M;do echo "-I gvcfs_chr${i}.vcf" ;done) \
    -O ./merge.vcf \
    1>./sample_log.GatherVcfs 2>&1
```

4.4 变异过滤

Filter Variants by VariantRecalibrator (VQSR)

由于评价SNP和Indel质量高低的标准是不同的，因此需要分别对SNP和Indel进行过滤，参数设置参考：[如何正确设置GATK VQSR的模型训练参数]

4.4.1 SNP 过滤

```
ref= ~/biodata/hg19.fa
vcf= ~/wgs/output/gatk/sample_HC.vcf  # 多样本为合并后的文件
~/wgs/output/gatk/merge.vcf
hampap= ~/biodata/hapmap_3.3.hg19.sites.vcf
omni= ~/biodata/1000G_omni2.5.hg19.sites.vcf
1000g= ~/biodata/1000G_phase1.snps.high_confidence.hg19.sites.vcf
dbsnp= ~/biodata/dbsnp_138.hg19.vcf
mill= ~/biodata/Mills_and_1000G_gold_standard.indels.hg19.sites.vcf

## VQSR
## 首先进行SNP过滤
time gatk --java-options "-Xmx20G -Djava.io.tmpdir=./tmp" VariantRecalibrator \
    -R $ref \
    -V $vcf \
    --resource:hapmap,known=false,training=true,truth=true,prior=15.0:${hampap}
\
    --resource:omni,known=false,training=true,truth=false,prior=12.0:${omni} \
    --resource:1000G,known=false,training=true,truth=false,prior=10.0:${1000g} \
    --resource:dbsnp,known=true,training=false,truth=false,prior=2.0:${dbsnp} \
    -an DP -an MQ -an QD -an FS -an SOR -an ReadPosRankSum -an MQRankSum \
    -mode SNP \
    -O sample_HC.snp.recal \
    --tranches-file sample_HC.snp.tranches \
    --rscript-file sample_HC.snp.plots.R \
    1>./sample_HC.snp.vqsr.log 2>&1

time gatk --java-options "-Xmx20G -Djava.io.tmpdir=./tmp" ApplyVQSR \
    -R $ref \
    -V $vcf \
    --truth-sensitivity-filter-level 99.0 \
    --tranches-file ./sample_HC.snp.tranches \
```

```
    --recal-file ./sample_HC.snp.recal \
    -mode SNP \
    -O ./sample_HC.snp.VQSR.vcf.gz \
    1>./sample_HC.snp.ApplyVQSR.log 2>&1 \
    && echo "** SNP ApplyVQSR done **"
```

4.4.2 Indels 过滤

```
## 之后进行Indel过滤 这里的input是上一步SNP VQSR后的vcf
time gatk --java-options "-Xmx20G -Djava.io.tmpdir=./tmp" VariantRecalibrator \
    -R $ref \
    -V ./sample_HC.snp.VQSR.vcf.gz \
    --resource:mills,known=false,training=true,truth=false,prior=12.0:${mill} \
    --resource:dbsnp,known=true,training=false,truth=false,prior=2.0:${dbsnp} \
    -an DP -an QD -an FS -an SOR -an ReadPosRankSum -an MQRankSum \
    -mode INDEL \
    -O sample_HC.snp.indel.recal \
    --tranches-file sample_HC.snp.indel.tranches \
    --rscript-file sample_HC.snp.indel.plots.R \
    1>./sample_HC.snp.indels.vqsr.log 2>&1

time gatk --java-options "-Xmx20G -Djava.io.tmpdir=./tmp" ApplyVQSR \
    -R $ref \
    -V ./sample_HC.snp.VQSR.vcf.gz \
    --truth-sensitivity-filter-level 99.0 \
    --tranches-file ./sample_HC.snp.indel.tranches \
    --recal-file ./sample_HC.snp.indel.recal \
    -mode INDEL \
    -O ./sample_HC.snp.indel.VQSR.vcf.gz \
    1>./sample_HC.snp.indel.ApplyVQSR.log 2>&1 \
    && echo "** Indels ApplyVQSR done **"
```

5. 变异注释

待定

ANNOVAR

汇总： 多样本批量处理脚本

0. 注意

```
## 工作目录
wgs/
├── bin
├── input
|   ├── sample1_1.fastq
|   ├── sample1_2.fastq
|   ├── sample2_1.fastq
|   ├── sample2_2.fastq
|   ├── ...
|   └── ...
└── output
    ├── align
    ├── gatk
    └── qc

## 参考基因组（已构建索引）和数据库文件
biodata/
├── gatk
|   └── bundle
|       └── hg19
|           ├── 1000G_omni2.5.hg19.sites.vcf
|           ├── 1000G_omni2.5.hg19.sites.vcf.idx
|           ├── 1000G_phase1.indels.hg19.sites.vcf
|           ├── 1000G_phase1.indels.hg19.sites.vcf.idx
|           ├── dbsnp_138.hg19.vcf
|           ├── dbsnp_138.hg19.vcf.idx
|           ├── hapmap_3.3.hg19.sites.vcf
```

```
|           ├── hapmap_3.3.hg19.sites.vcf.idx
|           ├── hg19.fa.gz
|           ├── Mills_and_1000G_gold_standard.indels.hg19.sites.vcf
|           └── Mills_and_1000G_gold_standard.indels.hg19.sites.vcf.idx
└── reference
    └── hg19
        ├── hg19.fa
        ├── hg19.fa.amb
        ├── hg19.fa.ann
        ├── hg19.fa.bwt
        ├── hg19.fa.dict
        ├── hg19.fa.fai
        ├── hg19.fa.gz
        ├── hg19.fa.pac
        ├── hg19.fa.sa
        └── nohup.out

## 在bin目录下创建一个config 文件，存放样本id，与fastq文件对应，方便后续批量处理
# 如sample1_1.fastq、sample1_2.fastq，对应的id为sample1

cat config
-------------------------
sample1
sample2
sample3
...
-------------------------

## 运行脚本时需进入环境
conda activate wgs

## 脚本均放在 ~/wgs/bin/ 目录下
```

1. 质控：fastqc.sh、trim_galore.sh

fastqc.sh

```
#!/bin/bash

# fastqc
cat config | while read id
do
    fastqc -o ../output/qc -t 16 ../input/${id}*.fastq.gz >>
../output/qc/${id}_fastqc.log 2>&1
done

# multiqc 合并结果
multiqc ../output/qc/*zip -o ../output/qc/multiqc
```

trim_galore.sh

```
## trim_galore.sh
cat config | while read id
do
    fq1=../input/${id}_1.fastq.gz
    fq2=../input/${id}_2.fastq.gz
    trim_galore --paired -q 28 --phred33 --length 30 --stringency 3 \
        --gzip --cores 8 -o ../output/qc $fq1 $fq2 >>
../output/qc/${id}_trim.log 2>&1
done
```

2. 比对：bwa.sh

```
#!/bin/bash
ref=~/biodata/reference/hg19/hg19.fa

cat config | while read id
do
    echo "Start: bwa for ${id}" `date`
    fq1=../output/qc/${id}_1_val_1.fq.gz  # 这里是trim_galore.sh 质控后的fastq文件
    fq2=../output/qc/${id}_2_val_2.fq.gz

    bwa mem -M -t 16 \
        -R "@RG\tID:${id}\tPL:illumina\tLB:WGS\tSM:${id}" \
        ${ref} ${fq1} ${fq2} \
        |samtools sort -@ 10 -m 4G -o ../output/align/${id}.bam \

    echo "Done: bwa for ${id}" `date`
done
```

3. GATK：标记去重 MarkDuplicates.sh

```
#!/bin/bash

cat config | while read id
do
    BAM=../output/align/${id}.bam

    # 判断是否已经生成~/wgs/output/gatk/ok.${id}_marked.status(即是否已经处理过该id的
bam文件),避免因程序中断，重新运行时重复提交代码
    if [ ! -f ../output/gatk/ok.${id}_marked.status ]
    then
        echo "Start: MarkDuplicates for ${id}" `date`
        ## GATK MarkDuplicates 标记或去除重复序列
        gatk --java-options "-Xmx20G -Djava.io.tmpdir=./tmp" MarkDuplicates \
            -I ${BAM} \
            --REMOVE_DUPLICATES true \
            -O ../output/gatk/${id}_marked.bam \
            -M ../output/gatk/${id}.metrics \
            1>../output/gatk/${id}_log.marked 2>&1 \

        # 运行结束，就创建一个ok.${id}_marked.status文件
        if [ $? -eq 0 ]
```

```
        then
            touch ../output/gatk/ok.${id}_marked.status
        fi

        echo "Done: MarkDuplicates for ${id}" `date`

        ## samtools 对生成的bam文件构建索引,这是后续步骤所需要的
        samtools index -@ 16 -m 4G -b ../output/gatk/${id}_marked.bam
../output/gatk/${id}_marked.bai
    fi
done
```

4. GATK：碱基质量重校正 bqsr.sh

```
#!/bin/bash

## 数据库文件和参考基因组
snp=~/biodata/gatk/bundle/hg19/dbsnp_138.hg19.vcf
indel=~/biodata/gatk/bundle/hg19/1000G_phase1.indels.hg19.sites.vcf
mills=~/biodata/gatk/bundle/hg19/Mills_and_1000G_gold_standard.indels.hg19.sites
.vcf
ref=~/biodata/reference/hg19/hg19.fa

cat config | while read id
do
    echo "Start: BQSR for ${id}" `date`

    ## GATK BaseRecalibrator 计算所有需重校正的reads和特征值,输出为一份校准表文件
    time gatk --java-options "-Xmx20G -Djava.io.tmpdir=./tmp"  BaseRecalibrator
\
        -R ${ref}  \
        -I ../output/gatk/${id}_marked.bam \
        --known-sites ${snp} \
        --known-sites ${indel} \
        --known-sites ${mills} \
        -O ../output/gatk/${id}_recal.table \
        1>../output/gatk/${id}_log.recal 2>&1 \

    ## GATK ApplyBQSR 利用校准表文件,重新调整原来BAM文件中的碱基质量值，并使用新的质量值重新
输出一份新的BAM文件
    time gatk --java-options "-Xmx20G -Djava.io.tmpdir=./tmp"  ApplyBQSR \
        -R ${ref} \
        -I ../output/gatk/${id}_marked.bam \
        -bqsr ../output/gatk/${id}_recal.table \
        -O ../output/gatk/${id}_bqsr.bam \
        1>../output/gatk/${id}_log.ApplyBQSR 2>&1

    ## samtools 对生成的sample_bqsr.bam文件构建索引,这是后续步骤所需要的
    time samtools index ../output/gatk/${id}_bqsr.bam

    echo "Done: BQSR for ${id}" `date`
done
```

5. GATK：变异检测 HaplotypeCaller.sh

```
#!/bin/bash

snp=~/biodata/gatk/bundle/hg19/dbsnp_138.hg19.vcf
ref=~/biodata/reference/hg19/hg19.fa

cat config | while read id
do
    bam=../output/gatk/${id}_bqsr.bam
    echo "Start: HaplotypeCaller for ${id}" `date`
    ## GATK HaplotypeCaller
    time gatk --java-options "-Xmx20G -Djava.io.tmpdir=./tmp" HaplotypeCaller \
        -R ${ref} \
        -I ${bam} \
        --dbsnp ${snp} \
        -O ../output/gatk/${id}_HC.vcf \
        1>../output/gatk/${id}_log.HC 2>&1
        # -ERC GVCF \ 单样本不需要

    echo "Done: HaplotypeCaller for ${id}" `date`
done
```

6. GATK：多样本合并 mergevcf.sh

```
#!/bin/bash
cd ../output/gatk
ref=~/biodata/reference/hg19/hg19.fa
for chr in chr{1..22} chrX chrY chrM
do
    ## GATK GenomicsDBImport
    time gatk --java-options "-Xmx20G -Djava.io.tmpdir=./" GenomicsDBImport \
        -R ${ref} \
        $(ls ./*HC.vcf | awk '{print "-V "$0" "}') \
        -L ${chr} \
        --genomicsdb-workspace-path gvcfs_${chr}.db \
    ## GATK GenotypeGVCFs
    time gatk --java-options "-Xmx20G -Djava.io.tmpdir=./tmp" GenotypeGVCFs \
        -R ${ref} \
        -V gendb://gvcfs_${chr}.db \
        -O ./gvcfs_${chr}.vcf \
done
## GATK GatherVcfs 合并所有染色体的vcf文件
time gatk --java-options "-Xmx20G -Djava.io.tmpdir=./tmp" GatherVcfs \
    $(for i in {1..22} X Y M;do echo "-I gvcfs_chr${i}.vcf" ;done) \
    -O ./cohort.vcf \
```

7. GATK：变异过滤 VQSR.sh

```
#!/bin/bash

cd ../output/gatk
```

```
ref=~/biodata/reference/hg19/hg19.fa
vcf=~/wgs/output/gatk/cohort.vcf  # 单样本为合并后的文件
~/wgs/output/gatk/${id}_HC.vcf
hampap=~/biodata/gatk/bundle/hg19/hapmap_3.3.hg19.sites.vcf
omni=~/biodata/gatk/bundle/hg19/1000G_omni2.5.hg19.sites.vcf
thousand=~/biodata/gatk/bundle/hg19/1000G_phase1.snps.high_confidence.hg19.sites
.vcf
dbsnp=~/biodata/gatk/bundle/hg19/dbsnp_138.hg19.vcf
mills=~/biodata/gatk/bundle/hg19/Mills_and_1000G_gold_standard.indels.hg19.sites
.vcf

## VQSR
## 首先进行SNP过滤
time gatk --java-options "-Xmx20G -Djava.io.tmpdir=./tmp" VariantRecalibrator \
    -R ${ref} \
    -V ${vcf} \
    --resource:hapmap,known=false,training=true,truth=true,prior=15.0 ${hampap}
\
    --resource:omni,known=false,training=true,truth=false,prior=12.0 ${omni} \
    --resource:1000G,known=false,training=true,truth=false,prior=10.0
${thousand} \
    --resource:dbsnp,known=true,training=false,truth=false,prior=2.0 ${dbsnp} \
    -an DP -an MQ -an QD -an FS -an SOR -an ReadPosRankSum -an MQRankSum \
    -mode SNP \
    -O cohort_HC.snp.recal \
    --tranches-file cohort_HC.snp.tranches \
    --rscript-file cohort_HC.snp.plots.R \
    1>./cohort_log.HC.snp.vqsr 2>&1

time gatk --java-options "-Xmx20G -Djava.io.tmpdir=./tmp" ApplyVQSR \
    -R ${ref} \
    -V ${vcf} \
    --truth-sensitivity-filter-level 99.0 \
    --tranches-file ./cohort_HC.snp.tranches \
    --recal-file ./cohort_HC.snp.recal \
    -mode SNP \
    -O ./cohort_HC.snp.VQSR.vcf.gz \
    1>./cohort_log.HC.snp.ApplyVQSR 2>&1 \
    && echo "** SNP ApplyVQSR done **"

## 之后进行Indel过滤 这里的input是上一步SNP VQSR后的vcf
time gatk --java-options "-Xmx20G -Djava.io.tmpdir=./tmp" VariantRecalibrator \
    -R ${ref} \
    -V ./cohort_HC.snp.VQSR.vcf.gz \
    --max-gaussians 4 \
    # resource里面参数要有 truth=true
    --resource:mills,known=false,training=true,truth=true,prior=12.0 ${mills} \

    --resource:dbsnp,known=true,training=false,truth=false,prior=2.0 ${dbsnp} \
    -an DP -an QD -an FS -an SOR -an ReadPosRankSum -an MQRankSum \
    -mode INDEL \
    -O cohort_HC.snp.indel.recal \
    --tranches-file cohort_HC.snp.indel.tranches \
    --rscript-file cohort_HC.snp.indel.plots.R \
    1>./cohort_log.HC.snp.indels.vqsr 2>&1

time gatk --java-options "-Xmx20G -Djava.io.tmpdir=./tmp" ApplyVQSR \
    -R ${ref} \
```

```
    -V ./cohort_HC.snp.VQSR.vcf.gz \
    --truth-sensitivity-filter-level 99.0 \
    --tranches-file ./cohort_HC.snp.indel.tranches \
    --recal-file ./cohort_HC.snp.indel.recal \
    -mode INDEL \
    -O ./cohort_HC.snp.indel.VQSR.vcf.gz \
    1>./cohort_log.HC.snp.indel.ApplyVQSR 2>&1 \
    && echo "** Indels ApplyVQSR done **"
```