第三次实验作业

12 APRIL 2021

> 实验背景
>
> > 由于人群中个体之间的基因组存在各式差异，参考基因组始终只能是参考，因此我们需要获取个体的基因组信息才能联系其表型来做有价值的分析；而测序成本的降低使得个人获取自己的全基因组信息变得现实，因此我们可以通过对癌症患者的种系细胞/体细胞测序，获取突变信息，来进行下游的分析如比较突变是否为遗传自父母。
>
> 实验方法
>
> > 通过获取的 TCGA 项目数据，以单个癌症患者种系细胞测序样本进行 GATK 流程[1]处理，鉴定分析样本基因组的点突变（SNV）情况。并以 `vcftool` 对变异结果进行简单分析。
>
> 运行环境（见下）
>
> ```
> platform          x86_64-pc-linux-gnu
> arch              x86_64
> os                linux-gnu
> system            x86_64, linux-gnu
> R version         4.0.5 (2021-03-31)
> nickname          Lost Library Book
> radian version    0.5.10
> python version    3.7.3
> ```

GATK 流程分析点突变情况

额外步骤：分割染色体

由于整个染色体较大，我们需要将其以合适的方法分割开，一种做法就是使用 `samtools`[2]将测序结果按照比对到的染色体分开，所用脚本如下。

```
#!/usr/bin/bash

# 指定每个任务分配的线程数，后同，不再重复注释
threadsPerTask=1

for i in {{1..22},X,Y,M}; do
    # 判断计算节点上任务是否超出用户上限，后同，不再重复注释
    while ((($(sq | wc -l) - 1) * threadsPerTask >= 40 || ($(sq | wc -l) - 1) >= 20))

        sleep 5
    done
    pkurun-cnlong 1 $threadsPerTask \
    # samtools view -b input.bam chrName > output_dedupped.bam
    "samtools view -b in/9086c9b5f797c7084060137f4b818525_gdc_realn.bam chr${i} > out/

done
```

`samtools view` 功能可以很方便地将 bam 文件按照染色体分割成单独的文件，在这里所有的文件名均为 `chr?.bam` 来表示比对到各个染色体的结果。

测序结果 QC 与校准 ——`MarkDuplicate`、`BaseRecalibrator`、`ApplyBQSR`

由于在测序过程中，一段 DNA 会被重复测到，因此会产生重复的片段。这样重复的片段是没有意义的，也不应该被当作支持或反对推定 SNV 存在的额外证据，因此在输出的 bam 文件中将会被标记为重复读段。所用脚本如下。

> GATK 管线在去重方面有更为严格的要求，但这样做的好处是使得我们可以分样本单独分析，以及更快的处理速度和并行计算的可能性。[3]

```
#!/usr/bin/bash

threadsPerTask=10

for i in {{1..22},X,Y,M}; do
    while ((($(sq | wc -l) - 1) * threadsPerTask >= 40 || ($(sq | wc -l) - 1) >= 20))

        sleep 5
    done
    pkurun-cnlong 1 $threadsPerTask \
        "gatk \
--java-options '-Xmx$((threadsPerTask * 3200))M -XX:+UseParallelGC -XX:ParallelGCThrea

MarkDuplicatesSpark \
-I out/chr${i}.bam \
-M out/Deduplicate_metrices_chr${i}.txt \
-O out/sorted_deduplicates_reads_chr${i}.bam"
done
```

其中由于 GATK 使用 java 编写，因此可以在调用时增添 `--java-options` 选项来指定最大内存和并行线程数等参数，此脚本中分配给每个染色体 10 个线程与 32G 内存。运行的结果会被保存在 `Deduplicate_metrices_` 开头的 txt 文件中，排序、去重后的 bam 文件以 `sorted_deduplicates_reads_` 开头。

碱基质量分是所有下游统计分析的基础，但由测序仪测出的碱基质量分往往不准确或带偏差，因此我们

需要通过已知的 SNP 位点来构建协方差模型，并以此来估计每个碱基的质量分。所用脚本如下。

```
#!/usr/bin/bash

threadsPerTask=1

for i in {{1..22},X,Y,M}; do
    while ((($(sq | wc -l) - 1) * threadsPerTask >= 40 || ($(sq | wc -l) - 1) >= 20))

        sleep 5
    done
    pkurun-cnlong 1 $threadsPerTask \
        "gatk \
--java-options '-Xmx$((threadsPerTask * 3200))M -XX:+UseParallelGC -XX:ParallelGCThrea

BaseRecalibrator \
-I out/sorted_deduplicates_reads_chr${i}.bam \
-R sharedResources/GRCh38.d1.vd1.fa \
--known-sites hg38/dbsnp_138.hg38.vcf.gz \
--known-sites hg38/Mills_and_1000G_gold_standard.indels.hg38.vcf.gz \
--known-sites hg38/1000G_phase1.snps.high_confidence.hg38.vcf.gz \
-O out/recalculated_data_chr${i}.table"
done
```

在这里我们使用一个人类全基因组参考序列作参考，三个已知位点的文件用于模型构建，输入上一步得到的去重后的 bam 文件，输出为一个观测值到实际值映射的表格，以 `recalculated_data_` 开头。我们可以借此对每个 bam 文件进行质量分的校准，所用脚本如下。

```
#!/usr/bin/bash

threadsPerTask=10

for i in {{1..22},X,Y,M}; do
    while ((($(sq | wc -l) - 1) * threadsPerTask >= 40 || ($(sq | wc -l) - 1) >= 20))

        sleep 5
    done
    pkurun-cnlong 1 $threadsPerTask \
        "gatk \
--java-options '-Xmx$((threadsPerTask * 3200))M -XX:+UseParallelGC -XX:ParallelGCThrea

ApplyBQSR \
-I out/sorted_deduplicates_reads_chr${i}.bam \
--bqsr-recal-file out/recalculated_data_chr${i}.table \
-R sharedResources/GRCh38.d1.vd1.fa \
-O out/recalibrated_chr${i}.bam"
done
```

使用 `--bqsr-recal-file` 参数指定上一步得到的表格，输入对应的 bam 文件与参考基因组，输出 `recalibrated_` 开头的碱基质量分经校准的 bam 文件。至此，我们得到了可以用与分析的 bam 文件。

鉴定二倍体生物的基因组变异位点 ——`HaplotypeCaller`、`CombineGVCFs`、`GenotypeGVCFs`

GATK 的 HaplotypeCaller 功能在遇到突变时，会放弃原有的比对结果，转而在局部对于读段进行重新匹配，因此可以同时鉴定 SNP 和 indel，输出的原始 gvcf 文件中也包含两种类型的突变。所用脚本如下。

```
#!/usr/bin/bash

threadsPerTask=4

for i in {{1..22},X,Y,M}; do
    while (((($(sq | wc -l) - 1) * threadsPerTask >= 40 || ($(sq | wc -l) - 1) >= 20))

        sleep 5
    done
    pkurun-cnlong 1 $threadsPerTask \
        "gatk \
--java-options '-Xmx$((threadsPerTask * 3200))M -XX:+UseParallelGC -XX:ParallelGCThrea

HaplotypeCaller \
-R sharedResources/GRCh38.d1.vd1.fa \
-L chr${i} \
--dbsnp hg38/dbsnp_138.hg38.vcf.gz \
-I out/recalibrated_chr${i}.bam \
--minimum-mapping-quality 30 \
-ERC GVCF \
-O out/chr${i}.g.vcf"
done
```

参数中指定了最低比对质量，用于粗筛原始突变结果，输出的文件格式指定为 gvcf。

由于我们先前将样本按染色体分成了多个样本，因此在执行下游的分析之前需要将 gvcf 文件合并为同一份（事实上即便是多样本的联合分析也需要将得到的 gvcf 进行合并），在这里我们将所有得到的 gvcf 文件整理为一个列表，并使用 GATK 的 CombineGVCFs 功能将列表中多个 gvcf 文件进行合并。所用脚本如下。

```
#!/usr/bin/bash

threadsPerTask=20

echo "" >chr.gvcf.list
for i in {{1..22},X,Y,M}; do
    echo "out/chr${i}.g.vcf" >>chr.gvcf.list
done
pkurun-cnlong 1 $threadsPerTask \
    "gatk \
--java-options '-Xmx$((threadsPerTask * 3200))M -XX:+UseParallelGC -XX:ParallelGCThrea

CombineGVCFs \
-R sharedResources/GRCh38.d1.vd1.fa \
--variant chr.gvcf.list \
-O out/combined.g.vcf && rm chr.gvcf.list"
```

在合并完 gvcf 文件之后我们将临时创建的列表删除，最终得到一个包含了全部样本变异信息的 `combined.g.vcf` 文件。

在得到了单个 gvcf 文件之后，我们使用 GATK 的 GenotypeGVCFs 功能进行联合基因分型注释。此功能接收的参数必须为单个 gvcf 文件，输出的文件中包含了分型后的结果，以 vcf 文件的形式呈现。

```
#!/usr/bin/bash

threadsPerTask=20

pkurun-cnlong 1 $threadsPerTask \
    "gatk \
--java-options '-Xmx$((threadsPerTask * 3200))M -XX:+UseParallelGC -XX:ParallelGCThrea

GenotypeGVCFs \
-R sharedResources/GRCh38.d1.vd1.fa \
-V out/combined.g.vcf \
-O out/raw.vcf"
```

至此，我们得到了包含了样本全部变异信息的 vcf 文件。

变异信息校准——`VariantRecalibrator`、`ApplyVQSR`

由于 GATK 具有非常高的灵敏度，因此难免会在原始结果中包含很多假阳性的突变信息，因此我们需要进行变异质量分的校准来弥补这一误差。所用脚本如下。

```
#!/usr/bin/bash

threadsPerTask=20

pkurun-cnlong 1 $threadsPerTask \
    "gatk \
--java-options '-Xmx$((threadsPerTask * 3200))M -XX:+UseParallelGC -XX:ParallelGCThrea

VariantRecalibrator \
-R sharedResources/GRCh38.d1.vd1.fa \
-V out/raw.vcf \
--resource:hapmap,known=false,training=true,truth=true,prior=15.0 hapmap_3.3.hg38.vcf.

--resource:omni,known=false,training=true,truth=false,prior=12.0 hg38/1000G_omni2.5.hg

--resource:1000G,known=false,training=true,truth=false,prior=10.0 hg38/1000G_phase1.sn

--resource:dbsnp,known=true,training=false,truth=false,prior=2.0 hg38/dbsnp_138.hg38.v

-an QD \
-an MQ \
-an MQRankSum \
-an ReadPosRankSum \
-an FS \
-an SOR \
-mode SNP \
-O out/variant_calibrated.recal \
--tranches-file out/variant_calibrated.tranches \
--rscript-file out/variant_calibrated.plots.R"
```

其中我们由于 SNP 和 indels 具有不同的特征模型，因此需要指明类型，用以训练校正模型。训练文件之后的参数指定了何种类型的注释会被用于训练模型，在这里我们指定了 vcf 文件中的测序深度（QD）、比对质量（MQ）、比对质量秩和（MQRankSum）、FisherStrand（FS）、StrandOddsRatio（SOR）参数用于训练模型，输出为一个用于校正变异质量的模型、一个包含了校对后变异质量分阈值的切片信息，以及一个用于绘图的 R 脚本，绘制的为模型不同维度下的变异分布，结果见此。

在建立了校准模型之后，我们可以使用 GATK 的 ApplyVQSR 功能来将之前得到的 vcf 文件依照阈值信息进一步注释，生成的 vcf 文件中每个变异均会被标注为 `PASS` 或 `FILTER`，并且仅为 SNP 类型的变

异进行矫正，所用脚本如下。

```
#!/usr/bin/bash

threadsPerTask=20

pkurun-cnlong 1 $threadsPerTask \
    "gatk \
--java-options '-Xmx$((threadsPerTask * 3200))M -XX:+UseParallelGC -XX:ParallelGCThrea

ApplyVQSR \
-R sharedResources/GRCh38.d1.vd1.fa \
-V out/raw.vcf \
-O out/variant_recalibrated.vcf \
-mode SNP \
--truth-sensitivity-filter-level 99.0 \
--tranches-file out/variant_calibrated.tranches \
--recal-file out/variant_calibrated.recal"
```

其中我们指定了 99.0 的过滤等级，意味着只保留 99% 的原始变异数据（一般推荐99% 或 99.9%，90% 会有最大的置信度）。

变异信息筛选——`SelectVariants`、`VariantFiltration`

在上一步对 SNP 类型的变异矫正过后，我们可以使用 SelectVariants 来提取出文件中所有的 SNP 类型突变。所用脚本如下。

```
#!/usr/bin/bash

threadsPerTask=20

pkurun-cnlong 1 $threadsPerTask \
    "gatk \
--java-options '-Xmx$((threadsPerTask * 3200))M -XX:+UseParallelGC -XX:ParallelGCThrea

SelectVariants \
-select-type SNP \
-V out/variant_recalibrated.vcf \
-O out/snp.vcf"
```

对于得到的 SNP 变异数据，我们可以使用 VariantFiltration 来进行一些硬性条件的过滤，比如可以使用 `--filter-expression` 指定一个或多个滤去的条件，并可用 `--filter-name` 指定保留的结果为滤过的或滤出的，最终得到一份高质量的 vcf 文件。所用脚本如下。

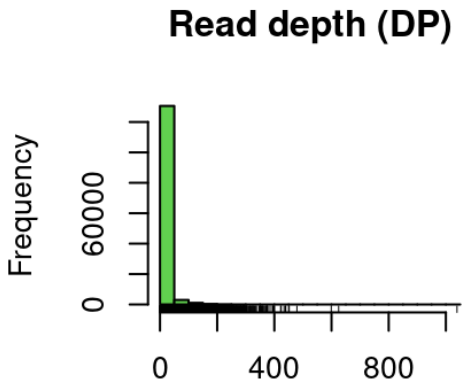
```
#!/usr/bin/bash

threadsPerTask=20

pkurun-cnlong 1 $threadsPerTask \
    "gatk \
--java-options '-Xmx$((threadsPerTask * 3200))M -XX:+UseParallelGC -XX:ParallelGCThrea

VariantFiltration \
-V out/snp.vcf \
--filter-expression \"QD<2.0||MQ<40.0||FS>60.0||SOR>3.0||MQRankSum<-12.5||ReadPosRankS

--filter-name \"PASS\" \
-O out/filtered.vcf"
```

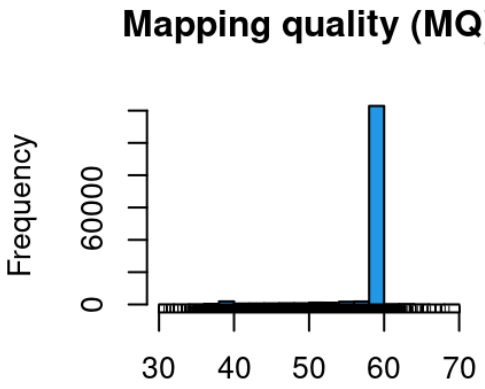
突变信息统计

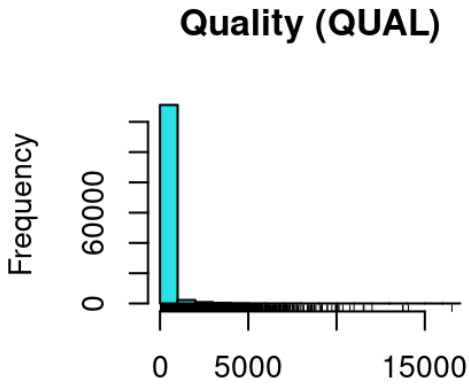
对于得到的 vcf 文件，我们可以使用 `vcfR` 软件包[4][5]进行作图分析，此处由于时间有限，故仅对第一条染色体上结果作做图展示（此软件包一次也仅能对一条染色体结果作图）。处理脚本与程序如下。

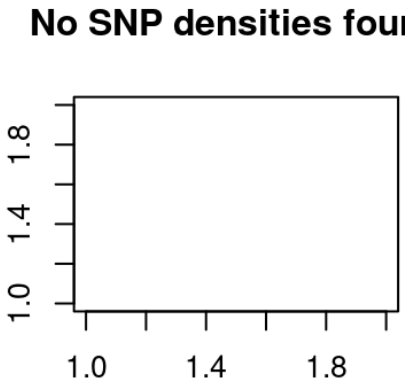
```
grep "^#" filtered.vcf > header.vcf
grep "^chr1[^0-9]" filtered.vcf > chr1.vcf
cat header.vcf chr1.vcf > filtered.chr1.vcf

grep "^#" gencode.v37.annotation.gff > header.gff
grep "^chr1[^0-9]" gencode.v37.annotation.gff > chr1.gff
cat header.gff chr1.gff > gencode.v37.annotation.chr1.gff
```

```
library(vcfR)
vcf_file <- "./filtered.chr1.vcf"
dna_file <- "./chr1.fa"
gff_file <- "./gencode.v37.annotation.chr1.gff"

vcf <- read.vcfR(vcf_file, verbose = FALSE)
dna <- ape::read.dna(dna_file, format = "fasta")
gff <- read.table(gff_file, sep = "\t", quote = "")

chrom <- create.chromR(
    name = "chr1",
    vcf = vcf,
    seq = dna,
    ann = gff
)

plot(chrom)
```

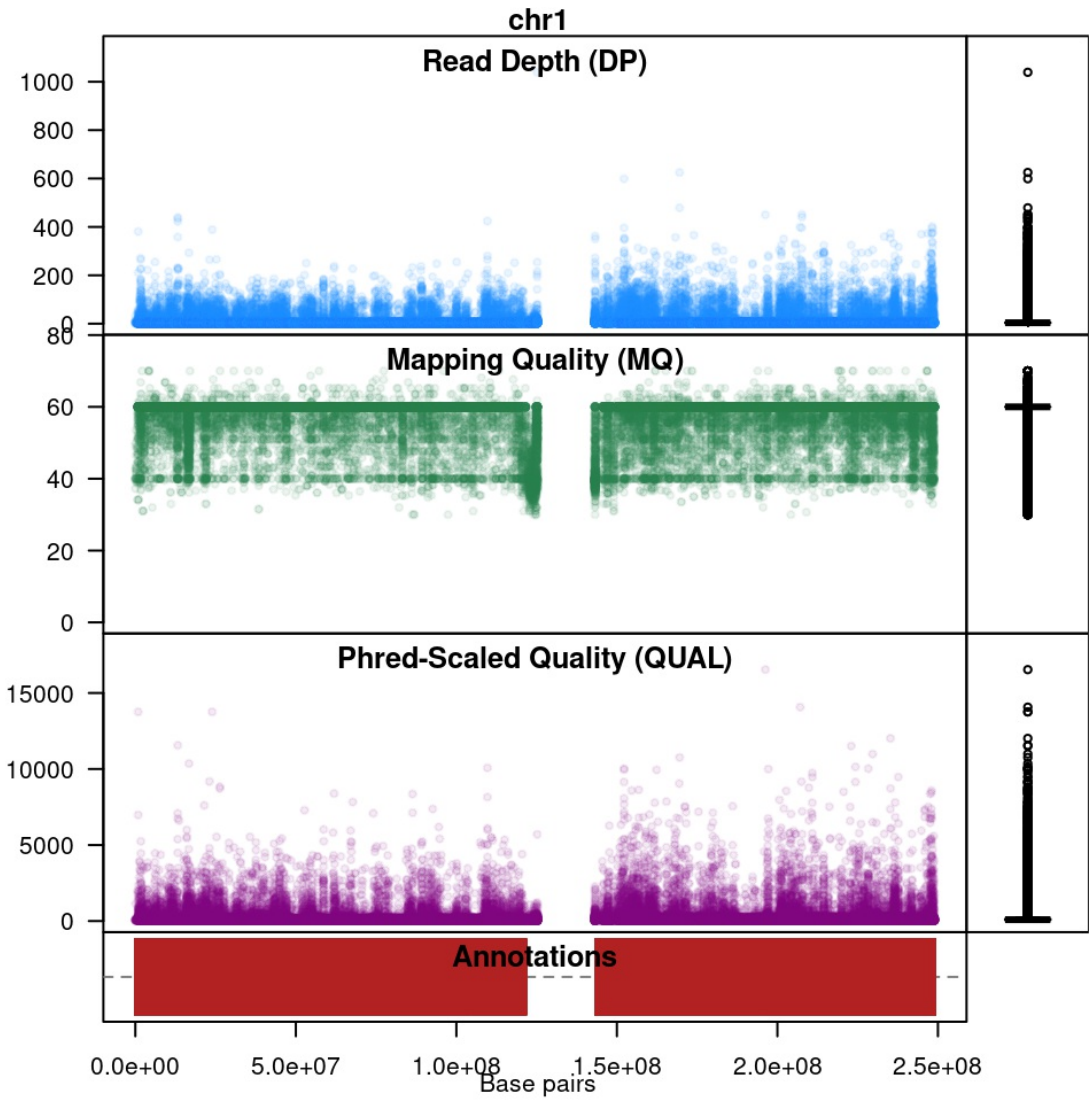









```
chromoqc(chrom, dp.alpha = 20)
```



参考资料

[1] MCKENNA A, HANNA M, BANKS E, 等. The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data[J]. Genome Res, 2010, 20(9): 1297–303. DOI:10.1101/gr.107524.110.

[2] DANECEK P, BONFIELD J K, LIDDLE J, 等. Twelve years of SAMtools and BCFtools[J]. GigaScience, 2021, 10(2). DOI:10.1093/gigascience/giab008.

[3] VAN DER AUWERA G A, CARNEIRO M O, HARTL C, 等. From FastQ data to high confidence variant calls: the Genome Analysis Toolkit best practices pipeline[J]. Curr Protoc Bioinformatics, 2013, 43(1110): 11.10.1–11.10.33. DOI:10.1002/0471250953.bi1110s43.

[4] KNAUS B J, GRÜNWALD N J. VCFR: a package to manipulate and visualize variant call format data in R[J/OL]. Molecular Ecology Resources, 2017, 17(1): 44–53. http://dx.doi.org/10.1111/1755-0998.12549.

[5] KNAUS B J, GRÜNWALD N J. VcfR: an R package to manipulate and visualize VCF format data[J/OL]. BioRxiv, 2016. http://dx.doi.org/10.1101/041277.