# NeuSomatic Work Flow

##### 安装

Github Link : <https://github.com/bioinform/neusomatic>

Python 2.7 and the following Python packages must be installed:

* pytorch >= 0.3.1
* Torchvision >= 0.2.0
* pybedtools >= 0.7.10
* pysam >=0.14.1
* zlib >=1.2.11
* numpy >=1.14.3
* scipy >=1.1.0
* biopython >=1.68

可以使用anaconda/miniconda安装或者pip

conda install zlib=1.2.11 numpy=1.14.3 scipy=1.1.0

conda install pytorch=0.3.1 torchvision=0.2.0 cuda80=1.0 -c pytorch

conda install cmake=3.12.1 -c conda-forge

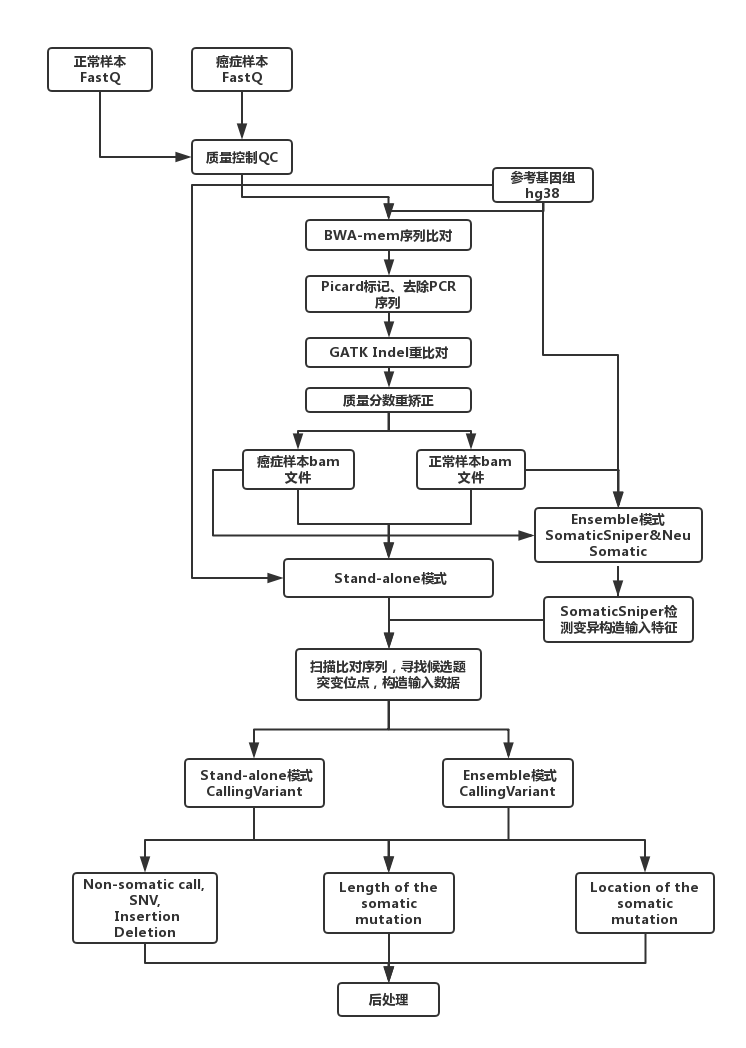
conda install pysam=0.14.1 pybedtools=0.7.10 samtools=1.7 tabix=0.2.5 bedtools=2.27.1 biopython=1.68 -c bioconda

编译：下载源码之后从neusomatic/bin 文件夹下运行 ./build.sh (需要cmake >=3.12.1 , g++ >=5.4.0).

编译过程中程序报错：

1. 看清错误内容，一般都是依赖问题，去安装对应的依赖文件。这一步必须一点错误都没有，否则运行过程中会报错！！！！

##### 步骤



####### 预处理：

**Note:以下所有包的使用都是按照流程顺序一步一步使用的，一般来说，某一步的输入是上一步的输出文件。**

1. # Trimmomatic去除测序质量低的序列和接头序列（分PE（pair end ）和SE(single end)模式）-phred33 -phred64代表测序质量，现在一般用33：

PE :

java -jar trimmomatic.jar PE -phred33 \

../WGS\_Work\_Flow/ReadsData/R1.fastq \ **## 输入**

../WGS\_Work\_Flow/ReadsData/R2.fastq \ **## 输入**

../WGS\_Work\_Flow/ReadsData/outR1.fastq.gz \ **## 输出**

../WGS\_Work\_Flow/ReadsData/outR1Trimm.fastq.gz \ **## 输出低质量序列(一般没用)**

../WGS\_Work\_Flow/ReadsData/outR2.fastq.gz \ **## 输出**

../WGS\_Work\_Flow/ReadsData/outR2Trimm.fastq.gz \ **## 输出低质量(一般没用)**

ILLUMINACLIP:adapters/TruSeq3-PE.fa:2:30:10 SLIDINGWINDOW:5:20 LEADING:5 TRAILING:5 MINLEN:50

1. # 建立人类参考基因组索引

bwa index HumenChromos.fasta

1. # 将经过质量控制的reads映射到基因组上并输出BAM文件

bwa mem -t 4 -R '@RG\tID:sra\_data\tPL:illumina\tLB:Non\tSM:humen' \ ##注意ID,LB,SM

~/WGS\_Work\_Flow/ChromeIndexFiles/GRCh38.fna \ ##人类参考基因组文件

~/WGS\_Work\_Flow/ReadsData/sra\_data.fastq **## 输入**

| samtools view -S -b - > Humen.bam ## **输出**并转换为bam文件

1. # 将BAM文件排序

samtools sort Humen.bam Humen\_sort.bam

1. # 删除重复序列

Samtools rmdup -S (PE model) Humen\_sort.bam **## 输入**Humen\_sort\_markdup.bam **## 输出**

或者

java -jar picard.jar MarkDuplicates \ ##删除重复序列命令

REMOVE\_DUPLICATES=true \ ## 直接在文件中移除重复序列

I=ReadsData/458QcMappingSort.bam \ ## **输入**数据

O=ReadsData/458QMSD.bam \ ## **输出**数据

M=ReadsData/458QMSDMeticx.txt ## 中间生成的矩阵

1. # 对最终BAM文件建立索引

samtools index Humen\_sort\_markdup.bam

1. # 为人类参考基因组fasta文件建立fai索引

samtools faidx data.fasta

1. # 对参考基因组创建dict文件

java -jar picard.jar CreateSequenceDictionary \ ## 命令

R=hg38/GRCh38.fasta \ ## 人类参考基因组

O=hg38/GRCh38.dict ## **输出**文件及其格式

1. # GATK局部重比对操作 (以NormalSample为例子，对TumorSample做一样的操作)

Note: -know / -knownSites代表着确定的位点信息，在局部重比对中可以不使用，但是不建议这么做。在碱基质量分数矫正的时候**必须使用**。在使用文件时**一定要注意已知的位点信息文件和参考基因组相匹配问题！！！**文件可在GATK网站上下载

第一步、

java -jar GenomeAnalysisTK.jar -T RealignerTargetCreator \ ## 命令

-R hg37/GRCh37.fasta \ ## 参考基因组序列

-I NormalSample.bam \ ## **输入**bam文件

-known /path/to/gatk/bundle/1000G\_phase1.indels.b37.vcf \

-known /path/to/gatk/bundle/Mills\_and\_1000G\_gold\_standard.indels.b37.vcf \

-o Normal.IndelRealigner.intervals ##**输出中间**文件

第二步、

java -jar GenomeAnalysisTK.jar

-T IndelRealigner \ ## 命令

-R hg37/GRCh37.fasta \ ## 参考基因组序列

-I NormalSample.bam \ ## **输入**bam文件

-o NormalMSDR.bam \ ## **输出**bam文件

-known /path/to/gatk/bundle/1000G\_phase1.indels.b37.vcf \

-known /path/to/gatk/bundle/Mills\_and\_1000G\_gold\_standard.indels.b37.vcf \

--targetIntervals Normal.IndelRealigner.intervals ## 上一步输出的中间文件

1. # GATK碱基质量分数矫正

第一步、

java -jar GATK/GenomeAnalysisTK.jar \ ## 命令

-T BaseRecalibrator \ ## 命令

-R hg19/hg19.fasta \ ## 参考基因组文件

-I NormalMSDI.bam \ ## **输入**bam文件

-knownSites hg19/1000G\_phase1.indels.hg19.vcf \

-knownSites hg19/Mills\_and\_1000G\_gold\_standard.indels.hg19.vcf \

-knownSites hg19/dbsnp\_138.hg19.vcf \

-o Normal.table ## **输出**中间文件

第二步、

java -jar GATK/GenomeAnalysisTK.jar -T PrintReads \ ## 命令

-R hg19/hg19.fasta \ ## 参考基因组文件

-I NormalMSDI.bam \ ## **输入**bam文件

-BQSR Normal.table \ ## 上一步**输出**的中间文件

-o NormalMSDIB.bam ## **最终输出**

1. # 将BAM文件排序

samtools sort NormalMSDIB.bam NormalMSDIBS.bam

1. # 打上MD标签并做index

samtools calmd -@ num\_threads -b alignment.bam reference.fasta > alignment.md.bam

samtools index alignment.md.bam

**NOTE:输入检测前必须有处理过的bam文件和对应的bai文件**

#### NeuSomatic 检测

## 第一步 ：

python neusomatic-master/neusomatic/python/preprocess.py --mode call \ ##命令

--reference hg19/hg19.fasta \ ## 参考基因组文件

--region neusomatic-master/resources/hg19.bed \ ## neusomatic自带的bed文件，在文件夹中有，**注意对应的参考序列版本**或者自己用对应的bed文件替换。

--tumor\_bam TumoralMSDIBSC.bam \ ## **输入**上游处理过后的肿瘤样本bam文件

--normal\_bam NormalMSDIBSC.bam \ ## **输入**上游处理过后的正常样本bam文件

--work work\_call \ ## **输出**这个是中间处理过程中会创建的文件夹，里面保留中间信息

--min\_mapq 10 \ ## 一次处理样本的最小数量

--scan\_alignments\_binary neusomatic-master/neusomatic/bin/scan\_alignments

**！！！！！特别注意，scan\_alignments文件是一个在linux环境下运行的2进制文件，文件是源码编译后得到的。这一步报错就是这个文件在编译过程中出现了问题，比如缺少依赖。如果在这一步出现错误一般需要从新从头开始编译文件！！！！！**

## 第二步、

python neusomatic-master/neusomatic/python/call.py ## 命令

--candidates\_tsv work\_call/dataset/\*/candidates\*.tsv ## **输入**上一步的work\_call文件夹中的文件

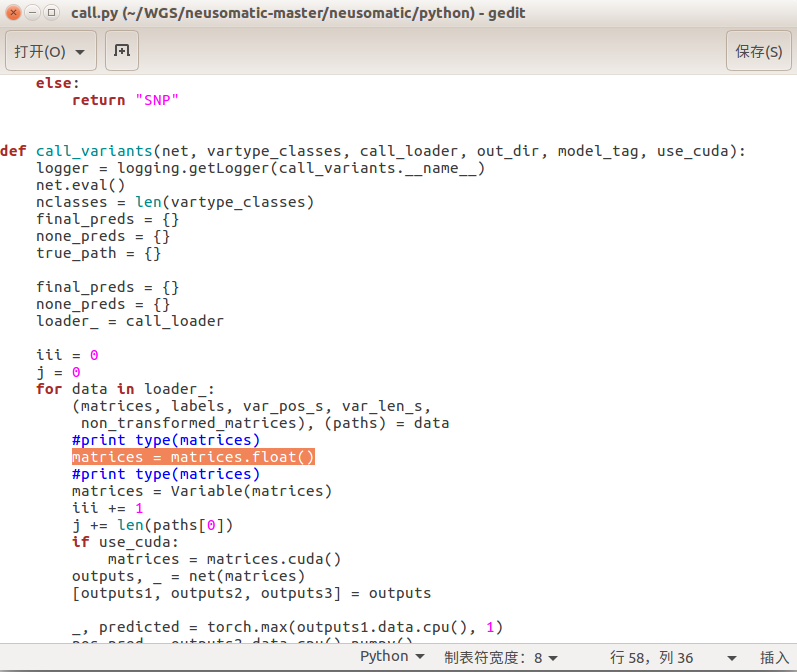
--reference hg19/hg19.fasta ## 参考基因组文件信息

--out work\_call ## **输出**检测结果输出目录

--checkpoint neusomatic-master/neusomatic/models/NeuSomatic\_v0.1.0\_standalone\_WEX\_100purity.pth ## pytorch模型参数文件

--batch\_size 100 ## 批量处理的数量

**！！！！！注意，这一步如果报错需要去call.py中修改源代码。这个是因为数据的格式在GPU或者CPU上是不一样的。英伟达GPU只支持单精度浮点数计算，这个模型的参数训练就是在GPU上进行的。所以在数据传入模型计算的时候需要改变数据类型，将双精度浮点数改为单精度！！！！！！具体修改如下：（在对应位置加入标记语句。）**



## 第三步、

python neusomatic-master/neusomatic/python/postprocess.py ## 命令

--reference hg19/hg19.fasta ## 参考基因组

--tumor\_bam TumoralMSDIBSC.bam ##**输入**肿瘤样本bam文件

--pred\_vcf work\_call/pred.vcf ## 预测的SNV（无用）

--candidates\_vcf work\_call/work\_tumor/filtered\_candidates.vcf ## 候选SNV（无用）

--output\_vcf work\_call/NeuSomaticThis.vcf ## **最终输出！！！这个是最终输出**

--work work\_call ## 前两步的work\_call文件夹

###### Training 和 Ensemble mode 由于没有数据所以并没有测试！！！！

GitHub教程地址 ：<https://github.com/bioinform/neusomatic>

###### 优缺点：

1. 可以使用自己的数据去训练模型，比较灵活。优点
2. 使用的特征数据较多。优点
3. 使用深度学习技术提高模型预测的准确性。优点
4. Bug太多，软件不友好。缺点