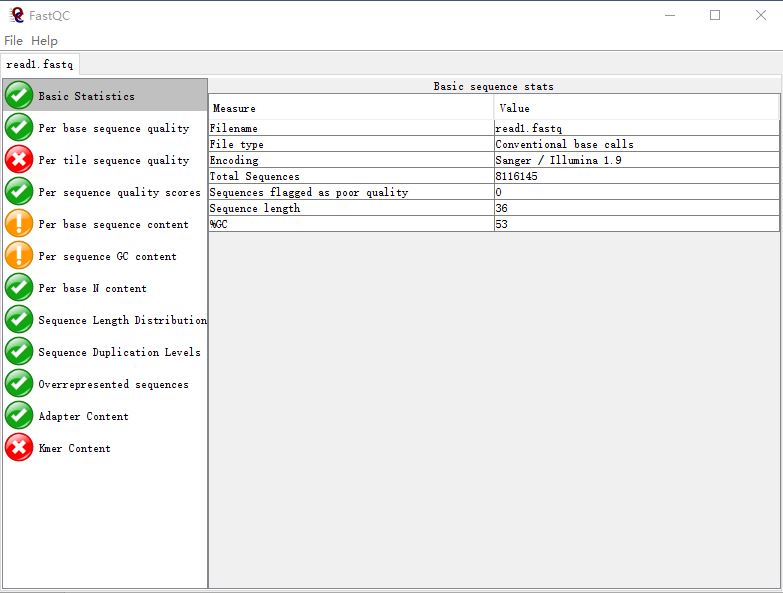
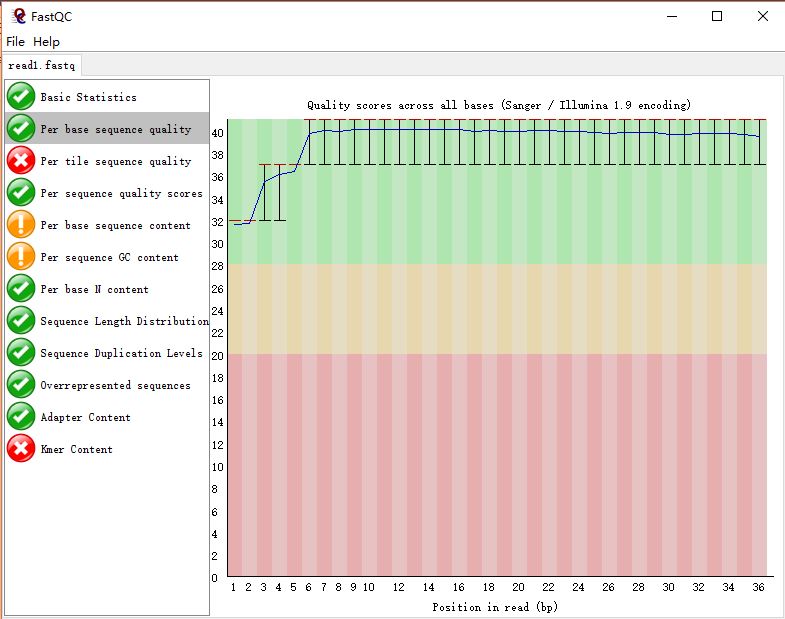
基因测序，从下机数据到分析结果！

一，质量控制

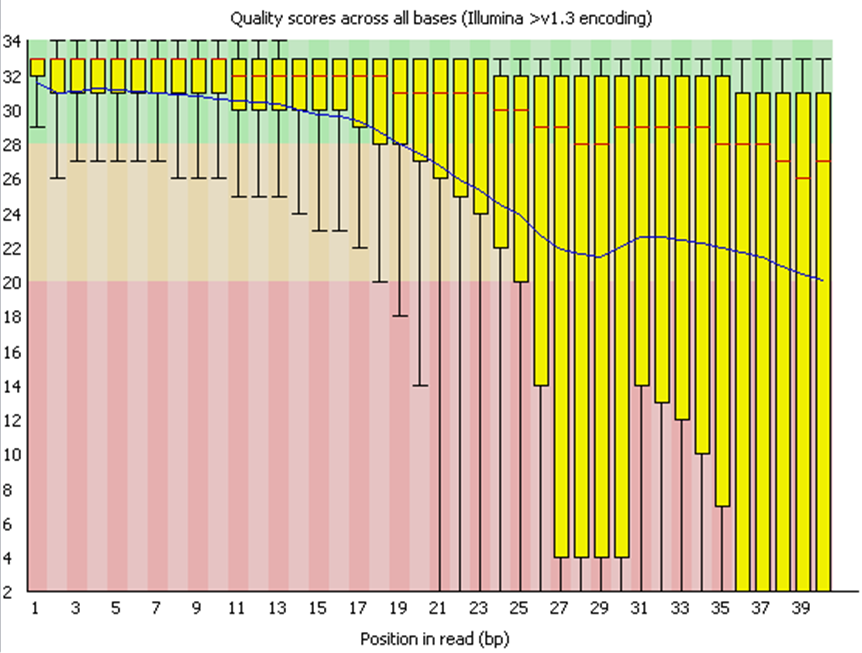
得到二代下机数据（fastq文件），首先要进行质量评估。用FastQC软件打开fastq文件，可以看到测序的信息。其中，TotalSequence为总长度。Sequencelength序列长度。总长=读长×reads数。所以，我们可用TotalSequence/Sequencelength得到reads数。



如图，能轻易计算出reads数大于22万。



由图可以看出这个数据质量挺好，是处理过的数据，并不是rawdata。一般来说，现在从测序公司拿到的数据都是这种处理过的。不会出现太杂乱的数据。



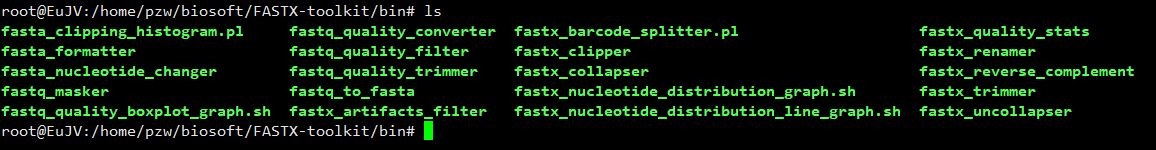
如上图就是没有处理过/质量差的数据的效果。

二，低质量数据过滤

数据已经处理过就不用进行这一步了！

要处理质量差的数据，主要的软件有

fastx toolkit。



主要用fastq\_quality\_trimmer和fastq\_quality\_filter。

$ fastx\_trimmer -h

usage:fastx\_trimmer [-h] [-f N] [-l N] [-z] [-v] [-i INFILE] [-o OUTFILE]

[-h] = This helpful help screen.

[-f N] = First base to keep. Default is 1 (=first base).

[-l N] = Last base to keep. Default is entire read.

[-z] = Compress output with GZIP.

[-iINFILE] = FASTA/Q input file. default is STDIN.

[-o OUTFILE] = FASTA/Q output file. default is STDOUT.

$ fastq\_quality\_filter -h

usage:fastq\_quality\_filter [-h] [-v] [-q N] [-p N] [-z] [-i INFILE] [-o OUTFILE]

[-h] = This helpful help screen.

[-q N] = Minimum quality score to keep.

[-p N] = Minimum percent of bases that must have [-q] quality.

[-z] = Compress output with GZIP.

[-iINFILE] = FASTA/Q input file. default is STDIN.

[-o OUTFILE] = FASTA/Q output file. default is STDOUT.

[-v] = Verbose - report number of sequences.

If [-o] is specified, report will be printed to STDOUT.

If [-o] is not specified (and output goes to STDOUT),

report will be printed to STDERR.

NGSQCToolkit比较简单

IlluQC.pl -pe read1.fastq read2.fastq 2 A -l 70 -s 20 -p 2 -z g

-pepaire-end数据

2 = Paired End DNA Library

A = Automatic detection of FASTQ variant

-l The cut-off value for percentage of read length that should be of given quality

-s参数 -cutOffQualScore read质量值cutoff

-z g=输出结果为压缩文件

三，比对至参考基因组

经典的软件就是bwa。

常用的数据库在这里<ftp://ftp.broadinstitute.org/bundle>。

一般是用hg19或者hg38。

首先要建立参考序列库。

bwa index ucsc.hg19.fasta

然后生成sam文件

bwa mem -M -t 1 ucsc.hg19.fasta read1.fastq.gz read2.fastq.gz | gzip -3 > readall.sam.gz

这里可以不解压，而且生成的sam直接用管道符压缩，节省空间。

四，Bam文件预处理

接下来就是用picard和samtools。其实就用picard也可以，因为picard基本能把samtools的事情（把sam文件转换为bam文件。Bam文件为二进制，运算更快）做完了。

1，对sam文件重新排序

（Picard的SortSam需指定一个tmp目录，用于存放中间文件，中间文件会很大，above 10G.注意指定目录的空间大小）。下面的命令顺便把sam文件转换为bam文件了，还按照coordinate排了序。（按染色体号排序）

Java -Xmx30g -jar picard.jar SortSam I=readall.sam.gz O=readall.bam SO=coordinate VALIDATION\_STRINGENCY=LENIENT TMP\_DIR='pwd'/tmp

2，去除这些由PCR扩增所形成的duplicates

在制备文库的过程中，由于PCR扩增过程中会存在一些偏差，也就是说有的序列会被过量扩增。这样，在比对的时候，这些过量扩增出来的完全相同的序列就会比对到基因组的相同位置。而这些过量扩增的reads并不是基因组自身固有序列，不能作为变异检测的证据，因此，要尽量去除这些由PCR扩增所形成的duplicates，这一步可以使用picard-tools来完成。去重复的过程是给这些序列设置一个flag以标志它们，方便GATK的识别。还可以设置 REMOVE\_DUPLICATES=true 来丢弃duplicated序列。对于是否选择标记或者删除，对结果应该没有什么影响，GATK官方流程里面给出的例子是仅做标记不删除。这里定义的重复序列是这样的：如果两条reads具有相同的长度而且比对到了基因组的同一位置，那么就认为这样的reads是由PCR扩增而来，就会被GATK标记。

java -Xmx6g -jar picard.jar MarkDuplicates I=readall.bam O=readall\_dedup.bam M=readall.metric VALIDATION\_STRINGENCY=LENIENT TMP\_DIR='pwd'/tmp

3，对bam文件进行加头（head）处理

GATK2.0以上版本将不再支持无头文件的变异检测。加头这一步可以在BWA比对的时候进行，通过-R参数的选择可以完成。如果在BWA比对期间没有选择-R参数，可以增加这一步骤。可使用picard-tools中AddOrReplaceReadGroups完成。

java -Xmx6g -jar picard.jar AddOrReplaceReadGroups I=readall\_dedup.bam O=readall\_final.bam RGID=group1 RGLB=lib1 RGPU=unit1 RGPL=illumina RGSM=readall VALIDATION\_STRINGENCY=LENIENT TMP\_DIR='pwd'/tmp

4，生成索引文件

java -Xmx6g -jar picard.jar BuildBamIndex I=readall\_final.bam O=readall\_final.bai TMP\_DIR='pwd'/tmp VALIDATION\_STRINGENCY=LENIENT

5，确认比对区域，减少错误

java -Xmx6g -jar GenomeAnalysisTK.jar -T RealignerTargetCreator -nt 1 -R ucsc.hg19.fasta -I readall\_final.bam -o readall.intervals -known Mills\_and\_1000G\_gold\_standard.indels.hg19.vcf

对于known sites的选择很重要，GATK中每一个用到known sites的工具对于known sites的使用都是不一样的，但是所有的都有一个共同目的，那就是分辨真实的变异位点和不可信的变异位点。如果不提供这些known sites的话，这些统计工具就会产生偏差，最后会严重影响结果的可信度。在这些需要知道known sites的工具里面，只有UnifiedGenotyper和HaplotypeCaller对known sites没有太严格的要求。

如果你所研究的对象是人类基因组的话，那就简单多了，因为GATK网站上对如何使用人类基因组的known sites做出了详细的说明，具体的选择方法如下表，这些文件都可以在GATK resource bundle中下载。

具体的选择方法：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 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 |  |  |  |  |  |

6，通过运行IndelRealigner在这些区域内进行重新比对

java -Xmx6g -jar GenomeAnalysisTK.jar -T IndelRealigner -R ucsc.hg19.fasta -I readall\_final.bam -targetIntervalsreadall.intervals -o readall\_realigned.bam -known Mills\_and\_1000G\_gold\_standard.indels.hg19.vcf

五，检测突变

将对输入的bam文件中的所有样本进行变异检测，最后生成一个vcf文件，vcf文件中会包含所有样本的变异位点和基因型信息。但是现在所得到的结果是最原始的、没有经过任何过滤和校正的Variants集合。这一步产生的变异位点会有很高的假阳性，尤其是indel，因此，必须要进行进一步的筛选过滤。这一步还可以指定对基因组的某一区域进行变异检测，只需要增加一个参数 -L：target\_interval.list，格式是bed格式文件。

java -jar -Xmx30g GenomeAnalysisTK.jar -T UnifiedGenotyper -R ucsc.hg19.fasta -L exome.bed -I readall\_final.bam -glm BOTH -stand\_emit\_conf 10 -stand\_call\_conf 30 -o noclean.vcf

六，位点初步筛选

java -Xmx15g-jar GenomeAnalysisTK.jar -R ucsc.hg19.fasta -T SelectVariants -V noclean.vcf -ef -o clean.vcf

七，注释突变

八，再次筛选

九，表型关联

十，表型匹配

十一，位点可视化

十二，一代验证