

NGS File Formats and Sequence Quality Check

Malay (malay@uab.edu)

November 14, 2016

Contents

1	NGS File formats	1
1.1	FASTQ	1
1.2	BAM or SAM format	1
1.3	VCF	2
2	Quality score	2
2.1	ASCII charater sets	2
2.2	Phred score	2
2.3	Why Phred 20	3
3	NGS Qaulity check	4
3.1	Fastqc	4
3.2	Read trimming	4

1 NGS File formats

1.1 FASTQ

A file format for getting the raw reads and the quality values. This is what you get from the sequencer. An example file can be found in the **ShortRead** package of BC (Figure 1).

1.2 BAM or SAM format

The FASTQ files are aligned against a reference genome using a software like BWA (<http://bio-bwa.sourceforge.net/>). The resulting alignment format is a BAM or SAM files. BAM files are binary, SAM files are plain text. The software for interconversion and analysis of these files are mainly **samtools** (<http://www.htslib.org/>). Sam file format specification can be found here <http://samtools.github.io/hts-specs/SAMv1.pdf>. A small example BAM files comes along with **Rsamtools** package (“example_from_SAM_Spec.sam”):

```
@ERR127302.8493430 HWI-EAS350_0441:1:34:16191:2123#0/1
GTCTGCTGTATCTGTGTCGGCTGTCTCGCGGGACATGAAGTCAATGAAGGCCTGGAATGTCACTACCCCAG
+
HHHHHHHHHHHHHHHHHHHHHEBDBB?B:BBGG<DDAA?AABFEFBD@DDECEE3>:?:@@@>?=BAB?##
@ERR127302.21406531 HWI-EAS350_0441:1:88:9330:2587#0/1
CTAGGGCAATCTTTGCAGCAATGAATGCCAATGGGTAGCCAGTGGCTTTTGAGGCCAGAGCAGACCTTCGGG
+
IIIIHHIIIGIIIIIIHHIIIEGBGHHIIHGHIIHHIIIIHHIIHHIIIIIGIIIEGIIGBGE@DDGGGIG
```

Figure 1: Example FastQ file.

```

@HD VN:1.3 SO:coordinate
@SQ SN:ref LN:45
r001 163 ref 7 30 8M2I4M1D3M = 37 39 TTAGATAAAGGATACTG *
r002 0 ref 9 30 3S6M1P1I4M * 0 0 AAAAGATAAGGATA *
r003 0 ref 9 30 5H6M * 0 0 AGCTAA * NM:i:1
r004 0 ref 16 30 6M14N5M * 0 0 ATAGCTTCAGC *
r003 16 ref 29 30 6H5M * 0 0 TAGGC * NM:i:0
r001 83 ref 37 30 9M = 7 -39 CAGCGCCAT *

```

1.3 VCF

Once the alignment BAM files have been generated, a variant caller like GATK (<https://www.broadinstitute.org/gatk/>) is used to find the variants in the file. The resulting file is called VCF. The specification can be found here (<http://samtools.github.io/hts-specs/VCFv4.2.pdf>). A sample VCF line is given below:

```

chr1 873762 . T G [CLIPPED] GT:AD:DP:GQ:PL 0/1:173,141:282:99:255,0,255
chr1 877664 rs3828047 A G [CLIPPED] GT:AD:DP:GQ:PL 1/1:0,105:94:99:255,255,0
chr1 899282 rs28548431 C T [CLIPPED] GT:AD:DP:GQ:PL 0/1:1,3:4:25.92:103,0,26

```

Once the variant is called they are annotated using variant annotation tools like SnpEff (<http://snpeff.sourceforge.net/>) or Annovar (<http://www.openbioinformatics.org/annovar/>) or **VariantAnnotation** package.

2 Quality score

2.1 ASCII character sets

Remember that to a computer everything is binary. The difference between a text file and a binary file is just to make the software reading the file to interpret the binary string representing “newline (\n)” differently than the other characters in the file.

Traditionally, each character in a computer is represented by 8 binary character or 1 byte. Although, 1 byte could represent 256 different characters only 128 used to get used. This character set is called ASCII. Then all the 256 possible code was included for representing english language. This character table is called **extended ASCII** or **latin1**. You can see the table just looking at the output of the command `man ascii`. Nowadays, due the demand to support non-english language, 256 characters possible by 1 byte is no longer enough, and a new standard for **multibyte** character set emerged, called unicode or UTF.

There are several versions of UTF. The simplest one is exactly like latin1 set called UTF+8. There are others, such as 2 byte UTF or UTF+16 or even 4 byte UTF called UTF+32. Which version a computer uses depends on **locale**. A software reading a file first looks at the locale and then interprets a text file based on the locale. If it is UTF+8, the software reads 1 byte at a time and then converts it into the character. For e.g., if the locale is UTF+8, and it finds the the following byte 01000001, it knows that this is the decimal 65 or the english character A.

2.2 Phred score

It is obvious from the discussion that any number can also be represented using character code. Using **Latin1** character sets, it is possible represent a number between 0 and 256. **Phred** originally used in a software called **phred** is a standard for representing quality score in a sequence file. **Phred** is a part of software package called **Staden**.

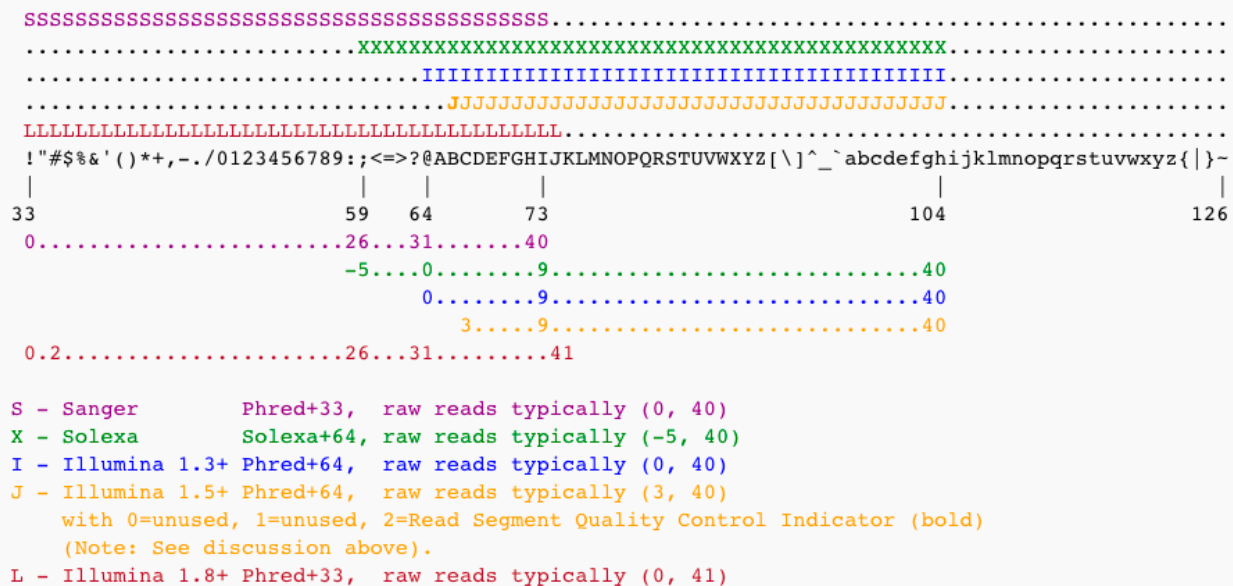


Figure 2: Phred score. (Source: wikipedia)

There used to be two types of Phred scores: Phred+33 and Phred+64. This number indicates the starting value. Nowadays Illumina sequencers generally used Phred+33 score.

Phred quality scores are defined as scaled logarithmic probability of an error in base-calling:

$$Q = -10 \log P$$

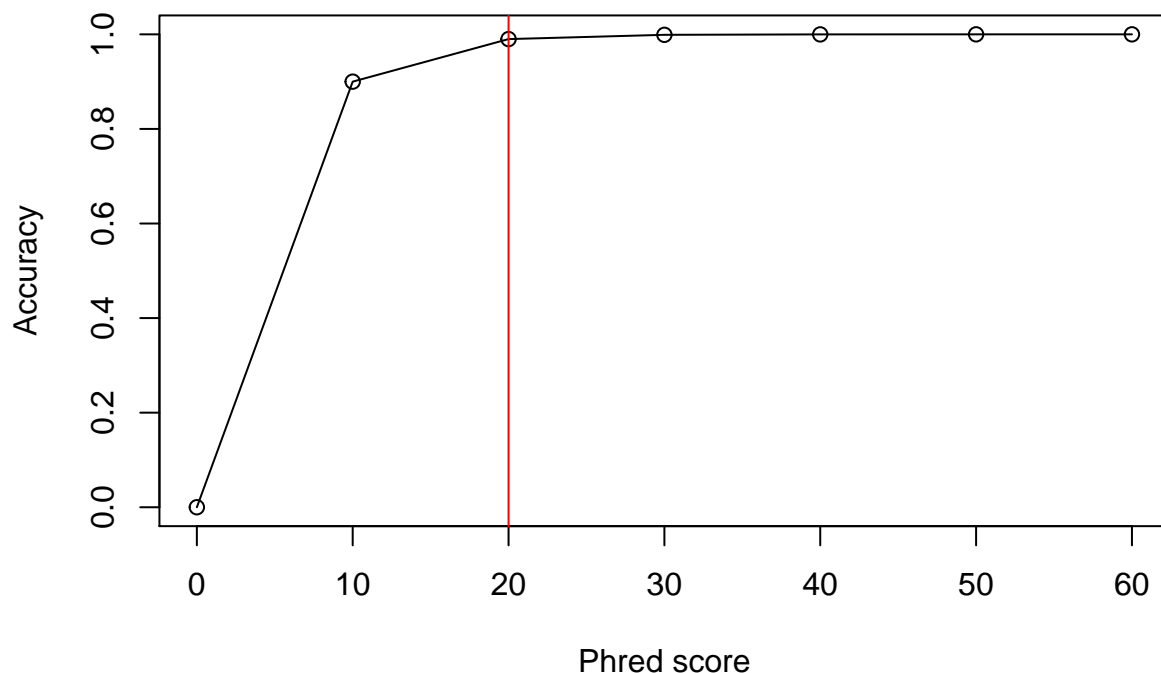
The number is then added to 33 to get the modern Phred+33 score. We can calculate the accuracy as follows:

$$accuracy = 1 - 10^{-Q/10}$$

2.3 Why Phred 20

A phred score of 20 is generally considered to be cutoff.

```
e <- seq(0,60,10)
a <- 1 - 10^(-(e/10))
plot(e,a,xlab="Phred score",ylab="Accuracy")
lines(e,a)
abline(v=20,col="red")
```



You can see there is sharp drop of quality below score 20. This is why Phred 20 is a good cutoff score. This actually $(20 + 33) = 53$ which 5 in ascii.

3 NGS Qaulity check

3.1 Fastqc

Fastqc is a program to check the quality of your file. Download Fastqc from here:

http://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc_v0.11.2.zip

An example of good file is http://www.bioinformatics.babraham.ac.uk/projects/fastqc/good_sequence_short_fastqc.html. An example of a bad file is http://www.bioinformatics.babraham.ac.uk/projects/fastqc/bad_sequence_fastqc.html. I suggest that you run **fastqc** from commandline. See **fastqc --help** for details.

3.2 Read trimming

We will use **Trimmomatic** for read trimming and adapter removal. Download Trimmomatic from:

<http://www.usadellab.org/cms/?page=trimmomatic>

Generally, Illumina adapters are of two types: Nextera for WGS and exome sequencing and Truseq from RNAseq. We need to keep that in mind and provide the for trimming.

```
java -jar Trimmomatic-0.33/trimmomatic-0.33.jar PE \  
D784G_R1.fq.gz D784G_R2.fq.gz \  
D784G_R1.trim_p.fq.gz D784G_R1.trim_u.fq.gz \  
D784G_R2.trim_p.fq.gz D784G_R2.trim_ufq.gz \  
ILLUMINACLIP:Trimmomatic-0.33/adapters/TruSeq3-PE.fa:2:30:10 \  
LEADING:2 \  
TRAILING:2 \  
SLIDINGWINDOW:4:15 \  
MINLEN:30
```

We used TruSeq3-PE adapters for clipping. Some other options are:

1. Adapters will have max 2 mismatches and will be clipped if a score of 30 is reached.
2. Remove leading and trailing N bases if quality is below 2.
3. Move with a 4 bp window and cutting where the average quality falls below 15.
4. After trmming remove all sequences whose length is below 30.