Whole-genome assembly
표준 프로토콜 (SOP)

2016.10.14

한양대학교 생명과학과
남진우 교수

해당 Whole-genome-assembly Standard Operation Protocol (SOP)은 Whole-genome-seq 을 이용하여 assembly 를 진행할 때의 기본 프로토콜을 목적으로 하여 설계되었음. 여기서 다루고 있는 Whole-genome-assembly 를 위한 시퀀싱 데이터는 Illumina 플랫폼의 paired-end 와 mate-pair 그리고 PacBio 플랫폼에 의해 생산된 long-read 시퀀싱 데이터를 대상으로 하고 있으며, 개별 연구자의 assembly 하려는 생물종, assembly 하는 목적에 따라 프로그램 별 세부 파라미터의 수정이 필요할 뿐만 아니라 시퀀싱 플랫폼 (플랫폼 별 error-rate)과 시퀀싱 디자인 (샘플 수, depth)에 따라 많은 변동사항이 있기 때문에 해당 SOP 에서 설명한 프로토콜은 모든 연구를 대변하는 방법론이 될 수 없음. 또한 현재 구축된 프로토콜은 최적화하는 단계를 통해 추후 변경될 가능성이 있음 (Whole-genome-assemly SOP 는 2016.10.14 현재 기준으로 작성되었음.).

목 차

1. 배경

Whole genome NGS data 는 최근 Illumina platform 이 표준으로 자리잡고 있으며 길이가 긴 PacBio data 를 이용한 assembly 방법들이 소개되고 있다. 정확도 측면에서는 PacBio data 만 이용해서 de novo assembly 를 수행하는 것이 최선이지만 최소 50X 이상의 sequencing depth 가 필요하여 sequencing 단계에서 상당히 많은 비용을 지불해야 한다. 따라서 경제적인 이유로 Illumina 와 PacBio read 를 혼합한 hybrid assembly 방법이 많이 소개되고 있다. 이 SOP 에서는 높은 depth 의 Illumina read 와 낮은 depth 의 PacBio read 를 이용한 hybrid assembly 방법을 소개한다 (그림 1).

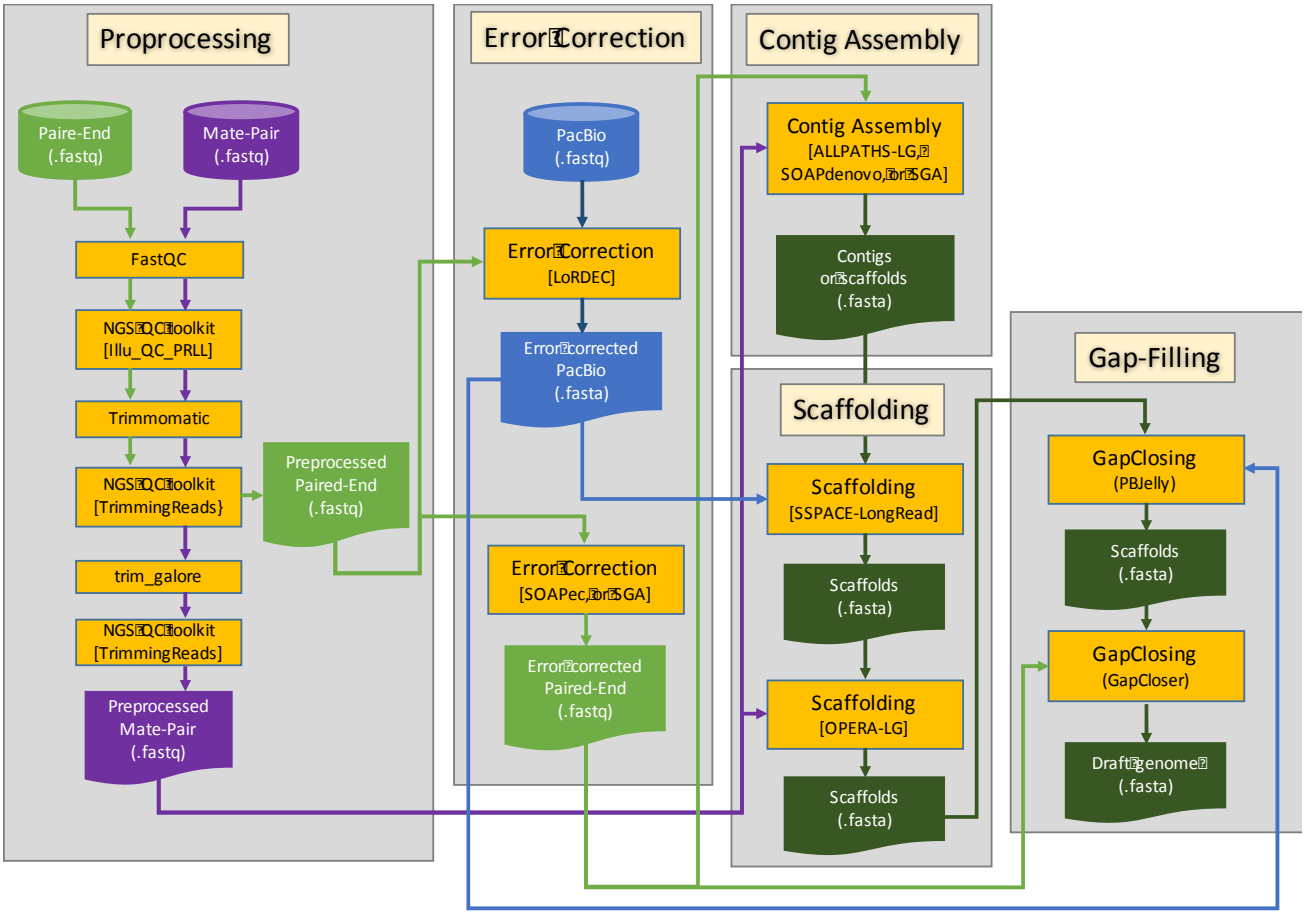

그림 1. Hybrid whole genome *de novo* assembly pipeline.

먼저, Illumina platform 으로 paired-end (PE) read 와 mate-pair (MP) read 를 생산한다. Paired-end read 의 길이는 100-300 base-pair (bp)의 forward-reverse 로 이루어진 쌍으로 구성되어 있으며 이 쌍의 양 끝단에 약간의 overlap 되는 구간이 존재한다. Mate-pair read 의 길이는 100-150bp 이고 insert-size 의 길이는 1-10 kilo-base-pair (Kbp)이며 그 길이는 library 구성에 따라 달라진다. PacBio read 의 per-base error 율을 약 13%로 매우 높기 때문에 이 역시 경우에 따라 사용하기 전에 sequencing error 를 correction 해야한다 (PacBio 를 이용한 scaffolding 에서는 굳이 correction 할 필요는 없다.).

Preprocessing 은 크게 quality control 및 trimming 단계와 sequencing error correction 단계로 나누어 볼 수 있으며 이후 단계는 크게 contig assembly, scaffolding, gap-closing 으로 나누어 볼 수 있다.

일반적인 whole genome *de novo* assembly 과정 (그림 2)

1) Preprocessing
   - Quality control 및 trimming
   - Sequencing error correction
2) Contig assembly
3) Scaffolding
4) Gap-filling

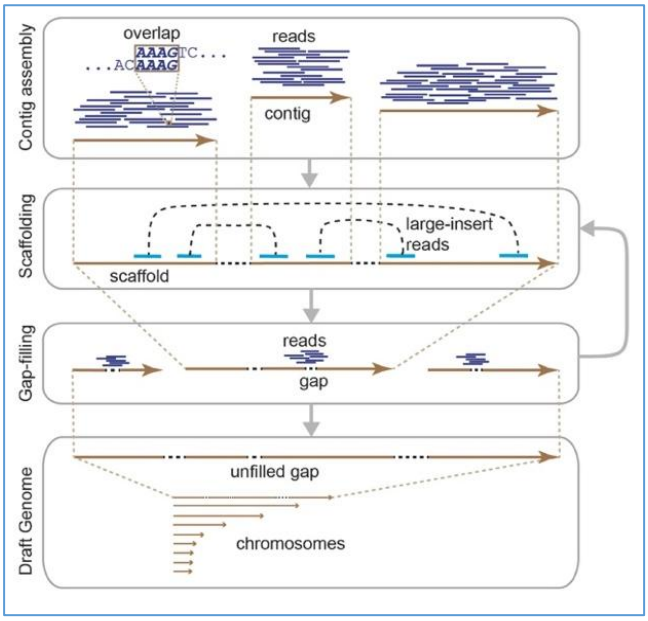

그림 2. The general workflow of whole genome *de novo* assembly.

Illumina PE/MP read 는 양 끝에 adaptor 나 vector 등에 의한 contamination 이 존재할 수 있고, sequencing error (<1%) 또한 존재하기 때문에 preprocessing 단계에서 반드시 이러한 contamination 과 sequencing error 를 제거해야 한다.

Contig assembly 단계에서는 PE read 를 이용해서 contig 를 생산하며 ALLPATHS-LG, SOAPdenovo, SGA 와 같은 프로그램이나 pipeline 들은 contig assembly 부터 gap-closing 까지의 단계를 모두 포함하고 있다.

Scaffolding 과정에서는 MP read 또는 PacBio read 를 contig 에 mapping 하여 그 정보를 이용해서 이미 assembly 된 contig 나 scaffold 들을 연결한다. 이때 contig/scaffold 간에 overlap 되는 구간이 존재한다면 하나로 합칠 수 있지만 그렇지 않은 경우에는 mapping 정보를 이용해서 contig/scafffold 간의 거리를 측정, 그 거리만큼 N 으로 채워진 gap 을 만들어 연결한다. MP read 를 이용해서 scaffold 하는 경우에는 read 의 길이가 짧기 때문에 mapping error 의 발생 빈도가 높고 그로 인한 mis-scaffolding 이 발생할 수 있다. 따라서 먼저 PacBio read 로 scaffolding 한 이후 MP read 로 scaffolding 하는 것이 좋다.

Scaffolding 과정에서 발생하는 gap 들 중 일부는 gap-closing 단계에서 PE read 또는 PacBio read 등을 이용해 메울 수 있다. 단, Illumina read 를 이용하는 경우에는 gap-closing 과정에서 많은 수의 mis-assembly 가 발생할 수 있다. 이러한 문제는 PacBio 를 이용한 gap-closing 을 먼저 수행함으로써 어느 정도 해결할 수 있다.

Scaffolding 과 gap-closing 단계는 더 이상의 scaffolding 이나 gap-closing 이 불가능할 때까지 반복적으로 수행할 수 있다. 이상적인 경우라면 그 결과물로써 chromosome 으로 이루어진 draft genome 을 얻을 수 있지만, 그러한 경우는 포유류와 같이 genome 크기가 큰 경우에는 거의 불가능하다. E.coli (genome size 4.6 mega-base-pair (Mbp)) 와 같이 genome size 가 작은 경우에 한해서 PacBio read 를 이용해서 chromosome 단위의 assembly 가 가능하다.

2. 준비

1) Whole genome sequencing data

- Illumina
  - Paired-end read: sequencing depth >20X
  - Mate-pair read: library 당 5-10X, insert size 2~10 Kbp (예, 3Kbp, 5Kbp, 10 Kbp)
- PacBio
  - sequencing depth >10X

2) 프로그램

- Quality control 및 trimming
  - NGS QC Toolkit[1], samtools[2], bamtools[3], Trimmomatic[4], FastQC[5]
- Sequencing error corrector
  - Illumina: SOAPec[6] 또는 SGA[7]
  - Illumina + PacBio: LoRDEC[8]
- *de novo* assembler
  - Illumina: ALLPATHS-LG[9], SOAPdenovo[6], SGA[7] 중 1 택
- Mapper
  - Illumina and PacBio: BWA-MEM[10], BOWTIE[11]
  - PacBio: BLASR[12]
- Scaffolder
  - PacBio: SSPACE-LongRead[13]
  - Illumina (Mate-pair) (+ PacBio): Opera-LG[14]
- gap-closer
  - PacBio: PBJelly[15]
  - Illumina: GapCloser[6]
- Quality assessment
  - QUAST[16]

3. Quality control 및 Trimming

- 작업을 시작하기에 앞서 assembly에 사용될 reads의 preprocess 작업이 필요함.
- Sequencing이 끝난 read는 아래와 같은 두 가지 문제점을 가질 수 있음.
  - Sequencing 과정 중에 base를 잘못 읽어서 생기는 error.
  - Adapter sequence나 bacteria sequence 등 외래 DNA부터 유래한 contamination.
- Preprocess의 목적은 최대한 이런 문제점을 제거하는 것임.

1) FastQC

- 내용
  - Fastq 파일로부터 데이터의 quality를 확인하기 위해서 실행함.
  - Quality score Q는 $Q = -10 \log P$ (Q = Phred quality scores, P = base-calling error probabilities) 의 공식으로 계산됨.
  - Preprocess의 각 단계마다 FastQC를 실행하여 중간 과정이 잘 실행되었는지 확인하면서 진행하여야 함.
  - FastQC 프로그램 자체적으로 data의 quality를 판단하여 Pass, Warning, Fail으로 표시하여 주지만 이는 일반적인 RNA-sequencing을 기준으로 하였을 때의 평가이며 data의 종류에 따라서 이 패턴을 따르지 않기도 함.

- 결과물

  1) fastqc_data1_1.txt : RNA-seq read 의 품질 정보가 쓰여진 파일.

```
##FastQC       0.10.1
>>Basic Statistics     pass
#Measure       Value
Filename       GSM546921_filtered_sequence.txt
File type      Conventional base calls
```

  2) fastqc_report.html: fastqc_data1_1.txt 의 내용을 웹 페이지로 볼 수 있는 파일.

  3) summary.txt 의 내용

```
PASS    Basic Statistics       GSM546921_filtered_sequence.txt
PASS    Per base sequence quality      GSM546921_filtered_sequence.txt
PASS    Per sequence quality scores    GSM546921_filtered_sequence.txt
WARN    Per base sequence content      GSM546921_filtered_sequence.txt
```

  fastqc_data1_1.txt 의 품질검사 단계에 따른 최종 결과 (PASS/WARN/FAIL)가 적힌 파일
  Per base sequence quality: read 의 각 base position 이 가지는 평균 quality score
  Per sequence quality scores: 각 read 가 가지는 평균 quality score 의 분포도
  Sequence Length Distribution: read 의 길이 분포도
  Sequence Duplication Levels: sequence 가 중복되는 reads 의 비율
  Overrepresented sequences: 예상보다 많이 관측되는 특정 Sequence

- 프로그램 다운로드
  http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

- 명령어의 예 (FastQC 프로그램)

```
$ fastqc [options] r1.fastq r2.fastq … rN.fastq [options]
```

  -o <output_dir>: output_dir를 지정해 주지 않으면 fastq 파일과 같은 폴더에 결과 파일 생성
  --casava: sequencing이 casava protocol을 따라서 진행되었다면 넣어야 함.

2) NGS QC toolkit [IlluQC_PRLL]

- 내용
  - Fastq파일로부터 Quality score가 낮은 read를 제거하기 위해 사용.
  - 대부분의 Quality control 프로그램들이 납득할 만한 quality score를 20으로 잡고 있으며 이는 1%의 error rate에 해당함.
  - 특히 genome assembly 과정에서는, genome 상에 위치하는 low complexity 영역에서 얻은 read의 경우 quality score가 상대적으로 낮을 수 있는데, 너무 높은 quality score를 cutoff로 주고 그 이상만을 남긴다면 위의 영역이 모두 사라질 수 있어서 20이 적당함.

- 프로그램 다운로드
  http://www.nipgr.res.in/ngsqctoolkit.html

- 결과물
  Seqfile1_filtered … seqfileN_filtered 이름을 가진 fastq 형식의 파일
  각 파일은 넣어준 매개변수의 기준을 충족하는 reads만을 가짐
  png파일들은 filtering 전 후의 reads들의 통계수치를 보여줌.

- 명령어의 예 (NGS QC toolkit 프로그램: IlluQC_PRLL)

```
$ perl IlluQC_PRLL.pl -pe r1.fastq r2.fastq N A -s 20 -l 70
```

  -pe: paired-end data 를 사용하기 위해서 입력함. single-end 라면 –se
  N: adapter library type 을 넣어주는 곳으로 adapter 는 다른 단계에서 제거할 예정이므로 아무것도 제거하지 않기 위해 N 을 입력함.
  A: FASTQ variants: A 는 Phred+33 인지 Phred+64 인지 프로그램 내에서 자동으로 판단해서 진행함.
  -s 20: cutOffQualScore 20 의 의미로 적어도 20 이상의 quality score 를 가지는 것을 기준으로 하겠다는 것을 의미함
  -l 70: read 전체 길이에서 70%가 cutOffQualScore 를 넘기지 못하는 read 는 제거됨.

3) Trimmomatic

- 내용
  - cDNA fragment를 읽기 위해서 adapter sequence를 붙이는데 이를 다시 제거하기 위해서 사용함.
  - Adapter sequence는 보통 5′ 그리고 3′에 각각 다른 sequence로 존재하며 이는 sequencing library 제작 과정에 따라서 달라짐.
  - Trimmomatic에서는 universal 하게 사용되는 일부 adapter sequence를 이미 제공하고 있어서 사용할 수 있음.

- 결과물

명령어에 넣어준 outputFile 이름을 가진 fastq 파일.

- 프로그램 다운로드
  http://www.usadellab.org/cms/index.php?page=trimmomatic

- 명령어의 예 (Trimmomatic 프로그램)

```
$ java -jar trimmomatic.jar PE r1.fastq r2.fastq r1_P.fastq \
r1_U.fastq r2_P.fastq r2_U.fastq \
ILLUMINACLIP:[PATH/TO/Trimmomatic]/adapters/TruSeq3-PE-2.fa:2:30:10 \
LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15 MINLEN:80
```

PE: paired-end data 를 사용하기 위해서 입력함. Single-end 라면 SE
r***1_P***.fastq r***1_U***.fastq r***2_P***.fastq r***2_U***.fastq: output data (순서에 유의할 것)
Note: r***1_U***.fastq 과 r***2_U***.fastq 는 이후 과정에서 사용하지 않는다.

ILLUMINACLIP:[PATH/TO/Trimmomatic]/adapters/TruSeq3-PE-2.fa:2:30:10 ₩
LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15 MINLEN:80

ILLUMINACLIP:3 : 제거하고자 하는 adapter sequence
LEADING:3 : 앞부분의 base quality 가 3 이하거나 N 이면 제거
TRAILING:3 : 뒷부분의 base quality 가 3 이하거나 N 이면 제거
SLIDINGWINDOW:4:15 : 4 base 씩 움직여가며 평균 quality 가 15 이하면 제거
MINLEN:80 : read 의 길이가 80 이하라면 제거

일반적인 Truseq adapter 를 사용한 경우 trimmer1 이 들어가야하는 위치에 위와 같은 명령어를 사용하면 됨.

4) NGS QC toolkit [TrimmingReads]

- 내용
  - Reads 말단 부분에서 quality score나 GC contents등 에서 일반적이지 않은 현상이 보이면 제거하기 위해서 사용.
  - Adapter가 제대로 제거되지 않았거나, sequencing 기술의 한계로 인해 reads의 말단 부분에 error가 많다면 5′ 또는 3′ 말단에서 일반적이지 않은 현상이 일어남.

- 프로그램 다운로드
  http://www.nipgr.res.in/ngsqctoolkit.html

- 결과물
  - OutputFile의 이름을 가진 fastq 파일이 생성됨.
    input으로 넣어준 fastq 파일에서 각각 왼쪽과 오른쪽에서부터 제거하고자 넣어준 길이만큼의 base를 제거된 fastq 파일.

- 명령어의 예 (NGS QC toolkit 프로그램: TrimmingReads)

```
$ perl TrimmingReads.pl -i forward.fastq -irev reverse.fastq -l <N> -r
<N> -o <outputFile>
```

-i forward.fastq: input file

-irev reverse.fasta: input file. forward.fastq 의 mate.

-l <N>: 왼쪽에서부터 제거할 base 의 숫자

-r <N>: 오른쪽에서부터 제거할 base 의 숫자

-o <outputFile> : outputFile 의 이름.

5) trim_galore

- 내용
  - Mate-pair로 생산한 reads에 붙어있는 NextEra adapter sequence를 제거하기 위해서 실행함.
- 프로그램 다운로드
  http://www.bioinformatics.babraham.ac.uk/projects/trim_galore/
- 결과물
  r1_val_1.fq r2_val_2.fq 이름의 fastq 파일과 report.txt 파일
  tirm_galore는 cutadapter를 사용하는 프로그램이기 때문에 report.txt에는 cutadapter의 결과 log가 적혀 있음.
- 명령어의 예 (trim galore 프로그램)

```
$trim_galore –paired --nextera r1.fastq r2.fastq
```

-paired: paired-end read 이기 때문에 입력함.

--nextera: nextera adapter sequence 를 제거하기 위해서 넣어 줌.

6) NGS QC toolkit [TrimmingReads]

- Mate-pair의 경우 NextEra adapter sequence까지 제거하고 나면 지나치게 짧은 길이의 read가 생성되기도 해서 이를 제거하기 위하여 사용.
- Read의 길이가 지나치게 짧으면 assembly 과정에서 mis-scaffolding을 일으킬 가능성이 커지기 때문에 read 길이가 63 이하면 제거함.
- 프로그램 다운로드
  http://www.nipgr.res.in/ngsqctoolkit.html
- 결과물
  - OutputFile의 이름을 가진 fastq 파일.
    넣어준 매개변수보다 긴 길이를 가지는 reads만 남아있는 fastq 파일.
- 명령어의 예 (NGS QC toolkit 프로그램: TrimmingReads)

```
$ perl TrimmingReads.pl -i forward.fastq -irev reverse.fastq -n 64 -o
<outputFile>
```

-i forward.fastq: input file

-irev reverse.fasta: input file. forward.fastq 의 mate.

-n 64: 64 보다 짧은 길이의 reads 는 모두 버림.

4. Sequencing error correction

1) Illumina reads error correction

- 일반적으로 *k*-mer 빈도수 히스토그램에서 *k*-mer depth가 낮은 *k*-mer들이 sequencing error를 포함하고 있다고 가정하고 error를 수정함.

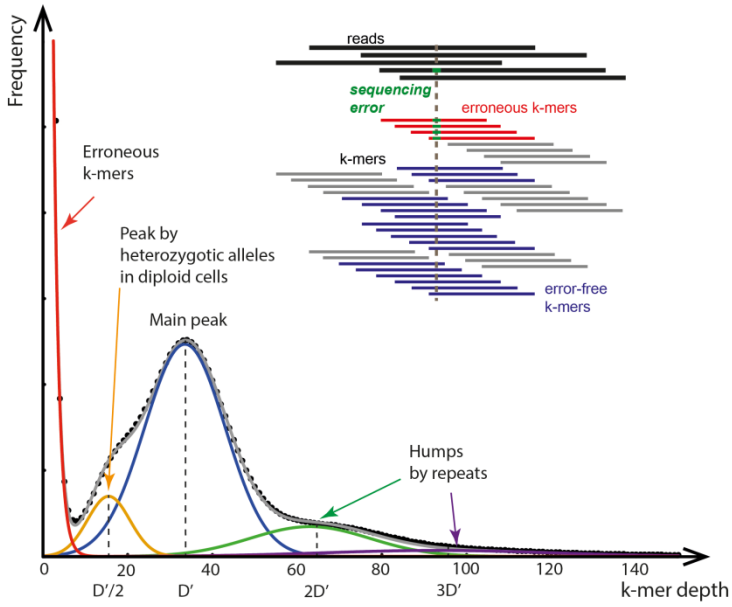

그림 3. The *k*-mer depth histogram

예를 들어, 그림 3에서 약 7 이하의 depth를 보이는 *k*-mer는 sequencing error로 가정함.

- Illumina read에 대한 sequencing error correction은 SOAPec 또는 SGA를 이용하고 PacBio error correction은 LoREC을 이용함.

2) SOAPec

- Illumina read의 error를 수정할 때 사용함.
- 프로그램 다운로드

  http://soap.genomics.org.cn/soapdenovo.html
- 입출력 데이터
  - Input: fastq 형식의 Illumina read data file. Gzip 파일 사용 가능

    (예: frag_1.fq.gz, frag_2.fastq)
  - output: fastq 형식의 Illumina read data file

    결과물: 입력 파일의 이름이 frag_1.fq, frag_2.fq인 경우 frag_*1*.cor.pair_*1*.fq.gz, frag_*2.cor*.pair_*2*.fq.gz 생성

- *k*-mer histogram 계산

```
$ KmerFreq_AR -k 17 -t 8 -p <prefix> -q 33 read.lst
```

k 17: *k*-mer 의 길이. 이 예에서는 17-mer 사용

-t 8: 사용할 CPU 개수. 이 예에서는 8 개 사용

-p <prefix>: output file 의 접두사.

-q 33: Phred score shift. 33 또는 64. Default 값이 64 이므로 반드시 확인할 것

read.lst: Illumina reads 를 기록하고 있는 file 에 대한 목록

예: read.lst 의 내용

frag1.pair1.fastq

frag1.pair2.fastq

...

- KmerFreq_AR 의 결과 file 을 이용해서 sequencing error 수정

```
$ Corrector_AR -k 17 -l 3 -Q 33 -t 8 <prefix>.freq.cz
<prefix>.freq.cz.len read.lst
```

-k 17: KmerFreq_AR 에서 사용한 *k*-mer 길이와 동일한 값 사용

-Q 33: KmerFreq_AR 에서 사용한 -q 옵션과 동일한 값 사용

-t 8: 사용할 CPU 개수. 이 예에서는 8 개 사용

-l: k-mer cut-off. 기본은 3 으로 설정되어 있으나 되도록이면 output.freq.cz.len 에서 확인 후 설정할 것

<prefix>.freq.cz, <prefix>.freq.cz.len: KmerFreq_AR 의 output file

3) SGA

- SGA에 내장된 error correction tool을 이용하여 sequencing error를 correction함.

- 프로그램 다운로드

https://github.com/jts/sga

- 입출력 데이터

- input: fastq 형식의 Illumina read file
- output: fastq 형식의 Illumina read file

- 명령어의 예 ((SGA): 41-mer, CPU 8 개를 사용하는 경우 paired-end read file 을 하나로 합함)

```
# SGA 의 preprocessing 과정으로서 paired-end read file 을 하나로 merge
$ sga preprocess --pe-mode 1 -o merged.fastq <pair1.fastq> <pair2.fastq>

# merge 된 파일을 인덱싱
$ sga index -a ropebwt -t 8 --no-reverse merged.fastq

# sequencing error correction
$ sga correct -k 41 --discard --learn -t 8 corrected.fastq
```

--pe-mode 1: Paired-end 이므로 1.

-a ropebwt: BWT algorithm 선택. long read의 경우 sais, Illumina의 경우 'ropebwt'

-t 8: 사용할 CPU 개수

--no-reverse: sequencing error를 위한 indexing에 반드시 필요

--discard: quality score가 낮은 read 배제

--learn: k-mer cut-off 자동 계산

Note: k-mer cut-off를 N으로 수동 설정할 경우 -x N 또는 --kmer-threshold=N 사용

corrected.fastq: 결과물 파일 이름

4) LoRDEC

- LoRDEC은 hybrid error correction tools로써 Illumina short read를 이용해서 PacBio read의 sequencing error를 correction하며 다른 프로그램에 비해 실행 속도와 성능이 뛰어남.

●프로그램 다운로드

http://www.atgc-montpellier.fr/lordec/

- 입출력 데이터
  - input: raw PacBio read와 sequencing error를 correction한 Illumina read
  - output: fasta 형식의 error corrected PacBio read

- 명령어의 예 (LoRDEC)

```
$ lordec-correct -i <PacBio> -2 <Illumina_list> -k <k-mer> -T 8 -o
<lordec.fasta> -s <k-mer_cut-off>
```

-i <PacBio>crror correction을 수행할 PacBio data file

-2 <Illumina_list>: Illumina read file에 대한 list

예:

fragment_pair1.fastq

fragment_pair2.fastq

-k <k-mer>: *k*-mer size

-T 8: 사용할 CPU 개수

-s <k-mer_cut-off>: cut-off 이하의 depth를 갖는 k-mer는 sequencing error로 취급, 배제.

-o <lordec.fasta>: 결과물 파일 이름

Note: fasta 형식으로 출력됨

- 결과물

```
# scaffold:
HUMAN/data/RUN/ASSEMBLIES/genome/final.assembly.fasta
# contigs:
HUMAN/data/RUN/ASSEMBLIES/genome/final.contigs.fasta
```

5. Contig assembly

- 시간과 RAM memory가 충분하다면 ALLPATHS-LG를 사용한다. ALLPATHS-LG는 현재까지 개발된 short read assembler 중에서 가장 긴 contig/scaffold를 생산하는 것으로 알려져 있음.
- 긴 contig나 scaffold 보다는 정확한 contig와 heterozygotic variant를 포함한 결과를 얻고자 하는 경우에는 SGA를 사용하는 것이 좋음.
- RAM memory가 충분하지 않거나 빠른 결과를 원하는 경우에는 SOAPdenovo를 이용하는 것이 좋음.

1) ALLPATHS-LG

- 내용
  - Sequencing error correction부터 contig assembly, scaffolding, gap-filling 등 whole genome de novo assembly의 모든 과정이 포함되어 있음.
  - Illumina read를 사용하는 assembler 중 가장 좋은 결과를 생산함.
  - 많은 양의 RAM memory가 필요하며 오랜 실행시간이 소요됨.
  - Human genome (3 Gbp)의 경우 최소 500 GB의 RAM memory 필요, 약 한 달 소요.
- 프로그램 다운로드
  http://software.broadinstitute.org/allpaths-lg/blog/
- 입출력 데이터
  - input: fastq 형식의 Illumina paired-end read와 mate-pair read (.fq.gz 형식 가능)
  - output: fasta 형식의 contig/scaffold
- 명령어의 예 (ALLPATHS-LG 프로그램)
  - PREPARE.sh 파일을 편집한다. (HUMAN assembly 의 경우)

```
PrepareAllPathsInputs.pl DATA_DIR=$PWD/HUMAN/data \
PLOIDY=2 \
IN_GROUPS_CSV=in_groups.csv \
IN_LIBS_CSV=in_libs.csv \
GENOME_SIZE=3100000000 \
```

  - ***in_groups.csv*** 내용 (comma-separated values). 필요한 경우 더 추가
  - ***in_libs.csv*** 내용 (comma-separated values). 필요한 경우 더 추가
  - genomic_start, genomic_end: 각 read 에서 사용 영역. 이 예에서는 jumping library (***SJ, LJ***)에서 5-65 만 사용.
    Note: in_libs.cvs 와 in_groups.cvs 의 Library name 을 동일하게 부여한다.(***FR, SJ, LJ***)

```
file_name,                library_name,  group_name
Fragment_Pair?.fastq,     FR,            FR-1
MatePair_2KB_Pair?.fastq, SJ,            SJ-1
MatePair_6KB_Pair?.fastq, LJ,            LJ-1
```

```
library_name, project_name, organism_name,    type, paired, frag_size, frag_stddev,
insert_size, insert_stddev, read_orientation, genomic_start, genomic_end
FR, Mammal, HUMAN, fragment, 1, 200, 40,     ,      ,  inward,  ,
SJ, Mammal, HUMAN, jumping,  1,   ,   , 2000,  400, outward, 5, 65
LJ, Mammal, HUMAN, jumping,  1,   ,   , 6000, 1200, outward, 5, 65
```

- Assembly 전 처리

```
$ makedir -p HUMAN
$ ./PREPARE.sh
```

- Ploidy file 생성: 이 파일이 없을 경우 실행도중 error 에 의해 프로그램이 정지할 수 있음.

```
$ touch HUMAN/data/ploidy
```

- RUN.sh 편집

```
RunAllPathsLG MAXPAR=2 \
THREADS=<사용할 CPU 갯수> \
PRE=${PWD} \
REFERENCE_NAME=HUMAN \      #data 가 저장될 directory 의 최상위 level
DATA_SUBDIR=data \          # HUMAN/data
RUN=RUN \                   # HUMAN/data/RUN
SUBDIR=genome \             # HUMAN/data/RUN/ASSEMBLY/genome
TARGETS=standard \
EVALUATION=NONE \
OVERWRITE=True \
MAX_MEMORY_GB=800 \
HAPLOIDIFY=True \
DRY_RUN=False | tee -a assembly.out
```

- RUN.sh 실행

```
$ RUN.sh
```

- 결과물

```
# scaffold:
HUMAN/data/RUN/ASSEMBLIES/genome/final.assembly.fasta
# contigs:
HUMAN/data/RUN/ASSEMBLIES/genome/final.contigs.fasta
```

2) SOAPdenovo

- 내용
  - ALLPATHS-LG에 비해 contig 및 scaffold 길이가 짧음.
  - 적은 양의 RAM memory를 요구하며 실행속도 역시 빠름.
  - 45-mer 사용시 ~100 GB, 1~2일 소요 (CPU 30개).
  - Sequencing error correction부터 gap-filling까지 모든 과정을 포함하고 있으며, 각 과정을 분리해서 사용할 수 있고 사용법이 간단함.

- 프로그램 다운로드
  http://soap.genomics.org.cn/soapdenovo.html

- 입출력 데이터

- input: sequencing error 를 correction한 fastq 형식의 Illumina paired-end read와 mate-pair read (.fq.gz 형식 가능)
- output: fasta 형식의 contig/scaffold

● 설정파일 편집

```
max_rd_len=101

[LIB]
avg_ins=200
reverse_seq=0
asm_flags=3
rank=1
q1=Frag1_Pair1.fastq
q2=Frag1_Pair2.fastq
q1=Frag2_Pair1.fastq
q2=Frag2_Pair2.fastq


[LIB]
avg_ins=3000
reverse_seq=1
asm_flags=2
rank=2
rd_len_cutoff=64
q1=MP-3kb_1.fastq
q2=MP-3kb_2.fastq
```

max_rd_len: 사용할 read 의 최대 길이
avg_ins: 평균 insert size 길이
reverse_seq: --> <-- 인 경우 (pair-end) 0, <-- --> 인 경우 (mate-pair) 1
asm_flags: contig assembly 에서만 사용할 경우 1,
scaffolding 에서만 사용할 경우 2,
contig, scaffolding, gap-filling 에서 모두 사용할 경우 3,
gap-filling 에서만 사용할 경우 4
rank: 사용 순서
q1, q2: pair 중 첫 번째는 q1, 두 번째는 q2
Note: 여러 library 를 사용하는 경우 [LIB] 추가

● SOAPdenovo 실행 (contig assembly부터 gap-closing까지의 모든 단계를 수행하는 경우)

```
$ SOAPdenovo-63mer all -s soap.conf -o HUMAN -K 45
```

-s soap.conf: SOAPdenovo 설정 파일
-o HUMAN: 결과 파일 이름의 접두사가 HUMAN인 경우
-K 45: *k*-mer size가 45인 경우
Note: 각각의 step을 독립적으로 실행하고자 하는 경우에는 all 대신 pregraph, contig, map, scaffold 등으로 교체하고 옵션을 조정해야 함. 자세한 내용은 다음 주소 웹페이지에서 확인.
http://soap.genomics.org.cn/soapdenovo.html

SOAPdenovo-63mer 또는 SOAPdenovo-63mer --help: SOAPdenovo 사용법

SOAPdenovo-63mer <command> 또는
SOAPdenovo-63mer <command> --help: SOAPdenovo command 사용법

- 결과물

```
- Scaffold 결과 파일: HUMAN.scafSeq
- Contig 결과 파일: HUMAN.contig
- Assembly 통계 파일: HUMAN.scafStatistics
```

3) SGA

- 내용
  - Overlap 기반의 어셈블러로서 *k*-mer를 이용해서 overlap 구간을 찾고 각 read를 직접 이어 붙이는 방식으로 assembly 함.
  - 45-mer 사용 시 ~100 GB, 1~2일 소요 (CPU 30개)
  - Contig 단위에서 매우 높은 정확도를 보임.
  - 추가적으로 heterozygotic variant 결과를 만들어 냄.
  - Segmental duplication이 유난히 많은 경우 다른 assembler들에 비해 좋은 결과를 생산함.

  Note: SGA는 contig assembly부터 scaffolding, gap-closing까지 모든 단계를 수행할 수 있지만 여기에서는 contig assembly만 설명함. Scaffolding은 SGA보다 OPERA의 결과가 더 좋다고 알려져 있음.

- 프로그램 다운로드
  https://github.com/jts/sga

- 입출력 데이터
  - input: sequencing error 를 correction한 fastq 형식의 Illumina paired-end read와 mate-pair read (.fq.gz 형식 가능)
  - output: fasta 형식의 contig/scaffold

- Contig assembly 실행

```
sga index -a ropebwt -t 30 corrected.fastq
sga filter -x 3 -t 30 corrected.fastq
sga fm-merge -m 55 -t 30 -o merged.fa corrected.filter.pass.fa
sga index -t 30 merged.fa
sga overlap -m 55 -t 30 merged.fa
sga assemble -m 55 -o NAME merged.asqg.gz
```

corrected.fastq: sequencing error 를 수정한 read.
-x 3: chimeric read 를 제거하기 위한 *k*-mer cut-off. 이 경우 3 사용
-t 30: 사용할 CPU 개수. 이 경우 30 개 사용
-m 55: pre-assembly 에서 필요한 최소 overlap. 이 예에서는 55 사용
-o NAME: 결과 파일의 접두사
sga 또는 sga --help: SGA 사용법 출력
sga <command> 또는 sga <command> --help: SGA 커맨드 사용법 출력

- 결과물

```
NAME-contigs.fa: contig assembly 결과
NAME-variants.fa: assembly 과정에서 발견된 heterozygotic variants
merged.asqg.gz: merged.fa 의 string graph 가 기록된 파일
```

6. Scaffolding

- 이미 assembly된 유전체의 scaffold N50 등을 높이거나, 혹은 minia assembler와 같이 scaffolding 과정이 없는 경우 Opera-LG, LINKS, SSPACE, SSPACE-LongRead 등의 scaffolder를 사용함.
  - 이 문서에서는 Opera-LG (for mate-pair)와 SSPACE-LongRead (for PacBio)를 이용한 scaffolding 과정을 설명함.
- 일반적으로 다양한 insert size의 mate-pair library를 이용하여 scaffolding하는 경우 가장 짧은 insert size를 보이는 것을 가장 먼저 사용함.
  - Opera(-LG)의 경우에는 모든 insert size의 library를 한꺼번에 사용하는 경우 더 정확한 좋은 결과를 생산한다고 알려져 있음.
- PacBio read와 Illumina Mate-Pair read를 모두 사용해서 scaffolding을 하는 경우에는 PacBio read를 먼저 사용할 때 더 좋은 quality의 assembly 결과를 얻을 수 있음.

1) SSPACE-LongRead

- 내용
  - PacBio long read를 이용한 scaffolding 프로그램으로써 Mate-Pair read를 이용한 방법보다 더 정확한 결과를 생산하고 실행 방법 또한 매우 간단함.
  - Running time은 contig 또는 scaffold의 개수와 유전체 내의 segmental duplication의 비율에 따라 달라질 수 있음. 1 ~ 7일 소요 (CPU 30개)

- 프로그램 다운로드
  http://www.baseclear.com/genomics/bioinformatics/basetools/SSPACE-longread

- 입출력 데이터
  - input: PacBio read (error correction 여부는 상관없으나 개발자는 raw PacBio를 추천함.)

- output: fasta 형식의 scaffold

● SSPACE-LongRead 실행

```
$ perl SSPACE-LongRead.pl -c <contig> -p <PacBio> -b <Output folder>
```

-c <contig>: scaffolding 대한 contig (또는 scaffold)

-p <PacBio>: PacBio read file 이름

-b <Output folder>: 결과 저장 위치 [default: PacBio_scaffolder_results]

-h 또는 --help: SSPACE-LongRead 사용법 출력

● 결과물

```
<Output folder>/scaffolds.fasta
```

1) Opera-LG

● 내용

- Mate-pair reads를 이용하여 scaffolding하는 데에 있어 가장 좋은 결과를 생산하는 scaffolder로 알려져 있다. 많은 BWA-MEM, BOWTIE 등의 alignment 프로그램으로 얻은 sam 형식의 file을 이용하여 scaffolding 함.
- 2 ~ 3일 소요됨.

● 프로그램 다운로드

https://sourceforge.net/projects/operasf/

● 입출력 데이터

- input: Illumina mate-pair read (옵션을 조정하면 paired-end 사용 가능).
- output: fasta 형식의 scaffold

Note: PacBio read도 입력 데이터로 사용가능 하지만 이 문서에서는 생략함.

● 명령어의 예 1

```
$ opera.sh <contig> <MatePair1> <MatePair2> <mapping file> [map_tool]
$ opera <contig> <mapping file list> <output-folder>
```

<contig>: assembly 된 contig (또는 scaffold)

<MatePair1> <MatePair2>: contig (또는 scaffold)에 mapping 할 mate-pair read 파일 이름

<mapping file>: .bam 형식으로 출력됨

[map_tool]: bwa 또는 bowtie (default bowtie)

mate pair library 가 여러 개 있을 경우 각각에 대하여 모두 수행

<mapping file list>: comma-seperated value (csv) 형식의 파일

예: MP-2kb.bam, MP-6kb.bam, MP-10kb.bam

Note: SAM file 도 사용 가능하다.

<output-folder>: 결과물을 저장할 directory

opera 또는 opera –h: OPERA 프로그램 사용법 출력

- 결과물

```
<output-folder>/scaffoldSeq.fasta        #scaffold 결과
<output-folder>/statistics               #결과에 대한 통계
```

- 명령어의 예 2: config file을 이용하는 경우. 더 자세한 설정이 가능하다.

```
$ opera config.txt
```

- config.txt 내용

```
output_folder=results
contig_file=contigs.fa

[LIB]
map_file=(directory)/Paired-End-300b.bam
cluster_threshold=5
lib_mean=300
lib_std=30
read_ori=in
```

cluster_threshold: threshold 이상의 pair 가 연결된 경우 하나의 cluster 로 묶음

lib_mean: 평균 insert size

lib_std: insert size 의 표준편차

read_ori: paired-end 나 fosmid read 의 경우 오리엔테이션에 주의

---> <--- : in

<--- ---> : out (default)

더 많은 library 가 존재하는 경우 [LIB] 추가

- 결과물

```
results/scaffoldSeq.fasta          #scaffold 결과
results/statistics                 #결과에 대한 통계
```

7. Gap-filling

- Illumina short read를 이용하는 경우IMAGE, GapFiller, Sealer, GapCloser 등의 프로그램을 이용하여 contig간에 존재하는 gap을 채울 수 있음. 그러나 short read를 이용하는 경우 매우 많은 mis-assembly를 양산할 수 있으므로 주의를 요함.
- Gap-filling 단계에서의 mis-assembly 문제를 완화하기 위해서는 PBJelly 등을 이용하여 PacBio read로 먼저 gap-filling을 수행하고 그 이후에 GapCloser등을 이용하여 short read로 gap-filling하는 것이 좋음.
- PacBio read를 이용하는 경우 PBJelly, GMcloser등을 이용하며 보다 정확한 gap-filling 결과를 기대할 수 있음.

1) PBJelly

- 내용

- PacBio read를 사용하기 때문에 Illumina read에서 커버하지 못하는 영역까지 gap-filling이 가능하다. 단, 16bp 이하의 gap은 low-quality 영역으로 간주하여 gap-filling을 하지 않음.
- BLASR를 mapper로 사용하기 때문에 다소 느림.
- Gap-closing 과정에서 PacBio read를 consensus를 실행하기 때문에 PacBio read의 sequencing error를 correction하지 않고 사용해도 무방함.
- 1 ~ 3주 소요

● 프로그램 다운로드

https://sourceforge.net/p/pb-jelly/wiki/Home/

● 입출력 데이터

- input: fastq/a 형식의 PacBio read (error correction 여부와 상관없이 사용가능.)
- output: fasta 형식의 scaffold/draft_genome

● protocol.xml 파일 편집

```
<jellyProtocol>
    <reference>scaffold.fasta</reference>
    <outputDir>PBJelly</outputDir>
    <blasr>-minMatch 8 -minPctIdentity 70 -bestn 1 -nCandidates 20 -
maxScore -500 -nproc 20 -noSplitSubreads</blasr>
    <input baseDir="./">
        <job>PacBio</job>
    </input>
</jellyProtocol>
```

nproc 20: mapper 인 blasr 가 사용할 CPU 갯수

Note: 다른 옵션은 default 로 사용하는 것을 권장함.

● PBJelly 실행

```
$ Jelly.py setup protocol.xml
$ Jelly.py mapping protocol.xml
$ Jelly.py support protocol.xml
$ Jelly.py extraction protocol.xml
$ Jelly.py assembly protocol.xml -x "--nproc=20"
$ Jelly.py output protocol.xml
```

--nproc=20: assembly 단계에서 사용할 CPU 갯수

`Jelly.py --help`: PBJelly 프로그램 사용법 출력

● 결과물

```
PBJelly/jelly.out.fasta
```

2) GapCloser

- 내용: Illumina read에 한해서 사용 가능함.
  - 2 ~ 7일 소요

- 프로그램 다운로드
  http://soap.genomics.org.cn/soapdenovo.html

- 입출력 데이터

  - input: error correction된 Illumina read
  - output: fasta 형식의 scaffold/draft_genome

- 명령어의 예 (GapCloser 프로그램)

```
$ GapCloser -a assembled.fasta -b read.lst -o output_file.fasta -t 30
```

  -a assembled.fasta: gap-filling 대상 scaffold 또는 draft genome
  -o output_file.fasta: 결과 파일 이름
  -t 30: 사용할 CPU 개수. 이 예에서는 30
  -b read.lst: Illumina read file 에 대한 list
  예:
  Frag1_1.fastq
  Frag1_2.fastq
  Frag2_1.fastq
  Frag2_2.fastq
  …
  Note: SOAPdenovo 를 이용하여 config file 의 asm_flag 를 4 로 조정한 후 실행해도 됨.

- 결과물

```
output_file.fasta
```

8. Quality assessment

- 참조 유전체 (reference genome)이 존재하는 경우에 한해 *de novo* assembly 결과에 대한 quality assessment를 할 수 있음.

1) QUAST

- 내용: Draft genome의 N50 통계 등 일반적인 metric과 함께 참조 유전체와 전사체를 이용하여 mis-assembly와 variation 등을 확인함.
  - 1 ~ 2 주 소요

- 프로그램 다운로드
  http://bioinf.spbau.ru/quast

- 입출력 데이터

- input: fasta 형식의 draft genome과 참조 유전체 파일, gff 형식의 참조 전사체 파일
  (ex) draft.fasta, reference.fasta, reference.gff
- output: report.txt를 포함한 각종 결과 파일

● 명령어의 예 (QUAST 프로그램)

```
$ python quast.py -R reference.fasta -G reference.gff3 -t 16 \
  draft.fasta
```

`-R reference.fasta`: 참조 유전체

`-G reference.gff`: annotation 된 참조 전사체

`-t 16`: 사용할 CPU 개수. 이 예에서는 16

`draft.fasta`: assembly 된 draft genome

`--help`: QUAST에 대한 자세한 옵션 설명 출력

● 결과물

```
# 결과물 저장 directory=(현재 dir)/quast_results/results_(날짜 및 시간)/
# 결과 파일
report.html
report.txt
report.pdf
...
aligned_stats/
contigs_reports/
contigs_reports/nucmer_output/
basic_starts/
genome_stats/
```

● `report.txt` 의 예 (다음 페이지)

All statistics are based on contigs of size >= 500 bp, unless otherwise noted (e.g., "# contigs (>= 0 bp)" and "Total length (>= 0 bp)" include all contigs).

Assembly	Draft_genome
# contigs (>= 0 bp)	1823
# contigs (>= 1000 bp)	1795
# contigs (>= 5000 bp)	531
# contigs (>= 10000 bp)	313
# contigs (>= 25000 bp)	122
# contigs (>= 50000 bp)	84
Total length (>= 0 bp)	1021077665
Total length (>= 1000 bp)	1021050359
Total length (>= 5000 bp)	1018557814
Total length (>= 10000 bp)	1016929008
Total length (>= 25000 bp)	1014081076
Total length (>= 50000 bp)	1012752244
# contigs	1823
Largest contig	196835485
Total length	1021077665
Reference length	1046932099
GC (%)	41.56
Reference GC (%)	34.10
N50	90110113
NG50	90110113
N75	29324116
NG75	23365841
L50	4
LG50	4
L75	9
LG75	10
# misassemblies	13106
# misassembled contigs	401
Misassembled contigs length	1014111047
# local misassemblies	15209
# unaligned contigs	54 + 393 part
Unaligned length	2532354
Genome fraction (%)	94.053
Duplication ratio	1.034
# N's per 100 kbp	1888.17
# mismatches per 100 kbp	505.72
# indels per 100 kbp	86.12
# genes	459 + 292 part
Largest alignment	2617572
NA50	404236
NGA50	395057
NA75	189637
NGA75	173519
LA50	710
LGA50	743
LA75	1629
LGA75	1736

9. References

1) http://www.nipgr.res.in/ngsqctoolkit.html
2) http://www.htslib.org
3) https://github.com/pezmaster31/bamtools
4) http://www.usadellab.org/cms/index.php?page=trimmomatic
5) http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
6) http://soap.genomics.org.cn/soapdenovo.html
7) https://github.com/jts/sga
8) http://www.atgc-montpellier.fr/lordec/
9) http://software.broadinstitute.org/allpaths-lg/blog/
10) http://bio-bwa.sourceforge.net
11) BOWTIE1: http://bowtie-bio.sourceforge.net/index.shtml,
    BOWTIE2: http://bowtie-bio.sourceforge.net/bowtie2/index.shtml
12) https://github.com/PacificBiosciences/blasr
13) http://www.baseclear.com/genomics/bioinformatics/basetools/SSPACE-longread
14) https://sourceforge.net/projects/operasf/
15) https://sourceforge.net/p/pb-jelly/wiki/Home/
16) http://bioinf.spbau.ru/quast