서열자료 조작을 위한 실용적인 스크립트 예제

작성자: 정해영 hyjeong@kribb.re.kr jeong0449@gmail.com (오류 정정이나 개선 사항에 대한 의견 환영!)
최초 작성일: 2015년 8월 6일
최종 수정일: 2015년 9월 22일
최초 공개일: 2015년 8월 7일

일러두기

- 이탤릭체로 쓴 것은 사용자가 상황에 맞게 입력해야 하는 곳을 나타냄
- 본 자료의 활용을 위해서는 리눅스 시스템에 EMBOSS 패키지와 samtools가 설치되어 있어야 합니다.

▶ Fasta/fastq 파일에 몇 개의 서열이 있는지 세어보기

```
$ grep -c  '>' multi_sequences.fasta
```

```
$ awk 'END {print NR/4}' file.fastq
```

※[awk]: 중괄호 앞뒤에 공백이 있어도 되고 없어도 된다. 즉, {*action*}이나 { *action* } 어느 형태를 써도 상관이 없다.

▶ Fastq 파일에서 앞부분의 read 4백만개(1600만 줄)만 뽑아서 별도의 파일로 저장하기

```
$ awk 'NR == 1, NR == 16000000' in_file.fastq > out_file.fastq
```

```
$ head -n 16000000 in_file.fastq > out_file.fastq
```

※[awk] 비교 연산자 앞뒤에 공백이 없어도 된다.

▶ Fastq 파일을 (single line) fasta 파일로 변환하기

Fasta file은 읽기 쉽도록 60 글자마다 줄바꿈을 하는 것이 일반적이다. single line fasta란 fastq 파일처럼 줄바꿈 없이 서열을 하나의 줄에 표현한 것과 같다. single line fasta로 전환하면 후속 작업이 편리해진다.

```
$ awk 'NR%4  == 1 {a=substr($0,2);} NR%4  == 2 {print ">" a "\n" $0}' file.fastq >
  file_single_line.fasta
```

※[awk]: 중괄호 {} 내부에 명령어가 하나만 들어가는 경우 세미콜론을 생략해도 무방하다. 가독성을 높이기 위하여 중괄호의 바로 내부에 공백을 넣어도 된다.

※[awk]: `{print ">" a "\n" $0}`과 `{print ">"a"\n"$0}`은 결과가 같다. print 명령어로 인쇄할 인자들 사이에 콤마를 찍으면 OFS 변수(=구분자, 기본값은 공백)를 사이에 넣는다. 만약 탭("\t")을 구분자로 설정하고 싶으면 awk -vOFS "\t" '*pattern* {*action*}' 형태로 실행하라.

```
$ sed -n '1~4{s/^@/>/;p}; 2~4p' file.fastq
```

▶ Multi-line fasta 파일을 single line fasta 파일로 변환하기

```
$ awk '!/^>/ { printf "%s", $0; n = "\n" } /^>/ { print n $0; n = "" } END { printf
  "%s", n }' file.fasta > file_single_line.fasta
```

▶ Single line fasta 파일을 다시 multi-line fasta 파일로 바꾸기

EMBOSS가 설치되어 있다면 seqret 유틸리티를 이용한다.

```
$ seqret -sequence single_line.fasta -outseq multi-line.fasta
```
※ -sequence와 -outseq는 생략하고 입출력 파일명만 순서에 맞게 제공해도 된다.

▶ Fasta 파일(single line) 내의 서열을 읽어서 ID와 길이를 출력하기

```
$ awk 'NR%2 == 1 {id=substr($0,2)} NR%2 == 0 {print id, " ", length($1)}' single_line.fa
```

길이에 따라 정렬하여 출력하려면 sort 명령어를 사용한다. 길이에 따른 내림차순으로 정렬(긴 서열이 앞으로 나오도록)하려면 sort -k 2 -nr 명령어를 실행한다.

```
$ awk 'NR%2 == 1{id=substr($0,2)} NR%2 == 0 {print id, " ", length($1)}' single_line.fa | sort -k 2 -n
```

▶ Fasta 파일(single line)읽어서 tab-delimited two-column file(ID와 서열)로 출력하기

```
$ awk -vOFS="\t" 'NR%2 == 1{id=substr($0,2)} NR%2 == 0 {print id, $1}' single_line.fa > out.txt
```

▶ ID와 서열로 이루어진 two-column file을 fasta file로 전환하기

```
$ awk -vOFS='' '{print ">",$1,"\n",$2;}' out.txt > file.fasta
```

▶ ID와 서열로 이루어진 two-column file에서 1 kb 이상 서열만을 선택하여 fasta 파일로 출력하기

```
$ awk 'length($2) > 1000 {print ">"$1"\n"$2}' file.tab > longerThan1kb.fasta
```

▶ Multi-fasta 파일 내의 서열들을 분리하여 각각 별도의 파일로 저장하기

```
$ awk '/^>/ {OUT=substr($0,2) ".fa"}; {print >> OUT; close(OUT)}' multi_sequences.fasta
```
또는 별첨의 fastaSplit.pl Perl 스크립트를 이용한다.

▶ Fasta 파일 내에서 특정 서열만 추출하기(ID 지정)
추출할 서열의 수가 많지 않아서 one-line script 내에 서열 ID를 삽입할 수 있는 경우

```
$ perl -ne 'if(/^>(\S+)/){$c=grep{/^$1$/}qw(id1 id2)}print if $c' sequences.fasta
```

서열 ID가 별도의 목록 파일에 들어있는 경우(한 줄에 ID 하나)

```
$ perl -ne 'if(/^>(\S+)/){$c=$i{$1}}$c?print:chomp;$i{$_}=1 if @ARGV' ids.txt sequences.fasta
```

▶ 1000 bp 이상의 서열만 추출하여 single line fasta 파일로 출력하기
Fasta/fastq 파일을 ID와 서열로 구성된 tab-delimited file로 먼저 전환해 둔다.

```
$ awk 'length($2) > 1000 {print ">"$1"\n"$2}' file.tab > longerThan1kb.fasta
```

▶ Fastq 파일로부터 70-120 bp 길이 범위의 read만 출력하기

Sequence quality는 고려하지 않는 단순한 방법이므로, quality를 기준으로 하는 고도한 trimming이나 filtering 등을 위해서는 전용 도구(예: FASTX-Toolkit, SolexaQA, prinseq 등)를 이용하는 것이 바람직하다.

```
$ awk 'BEGIN {OFS = "\n"} {header = $0 ; getline seq ; getline qheader ; getline qseq
  ; if  (length(seq)  >=  70  &&  length(seq)  <=  120)  {print  header,  seq,  qheader,
  qseq}}' infile.fastq > filtered.fastq
```

▶ 하나의 interleaved fastq file을 2개의 paired file로 분리하기

```
$ awk 'BEGIN{OFS="\n"} $0~/\/1$/{header = $0; getline seq; getline qheader; getline
  qseq; print header, seq, qheader, qseq}' infile.fastq > file_1.fastq
$ awk 'BEGIN{OFS="\n"} $0~/\/2$/{header = $0; getline seq; getline qheader; getline
  qseq; print header, seq, qheader, qseq}' infile.fastq > file_2.fastq
```

서열의 헤더에서 /1 /2와 같은 read의 pair 표시가 어떻게 되어 있는지 반드시 미리 확인하여 이에 맞는 awk 구문 내 패턴을 결정해야 한다.

@HWI-ST908R:151:D1M9MACXX:1:1101:1373:2469 2:N:0:CCGTCC => $2~/2:/

@BYI6V:00004:00005/1 => $0~/\/1$/

sed를 사용하면 한층 더 간단하게 작업할 수 있다. 서열의 ID를 실제로 확인하여 패턴을 결정해야 한다.

```
$ sed -n '/1:N/ {N;N;N;p;}' infile.fastq > file_1.fastq
$ sed -n '/2:N/ {N;N;N;p;}' infile.fastq > file_2.fastq
```

▶ Fasta file에서 특정 영역을 추출하기

먼저 EMBOSS의 seqret을 이용하는 방법을 알아보자. 추출할 서열의 시작 위치는 항상 끝 위치보다 앞에 있어야 한다. 따라서 reverse strand에 암호화된 유전자 염기서열을 추출할 때에도 stop codon에 해당하는 위치가 시작 위치로 주어져야 하며, -sreverse 옵션으로 상보 서열 전환을 하면 된다.

```
$ seqret -sbegin 20 -send 200 single_sequence.fa out.fa
$ seqret -sbegin 20 -send 200 single_sequence.fa genbank:out.fa
$ seqret -sbegin 20 -send 200 multi_sequences.fa:seq_ID out.fa
$ seqret -sbegin 20 -send 200 -sreverse single_sequence.fa out.fa
```

seqret이 출력하는 서열의 ID는 입력물의 것을 그대로 가져다 쓰게 된다. 출력 서열의 ID를 원하는 대로 수정하려면 실행 시간이 좀 더 소요되지만 sed를 사용하여 '>'로 시작하는 서열 ID 라인을 수정하면 된다.

```
$  seqret  -sbegin  100  -send  200  -stdout  -auto  multi_sequences.fa:seqID  |
  's/^>.*$/>seqID:100-200/' > out.fa      출력물: >seqID:100-200
$  seqret  -sbegin  100  -send  200  -stdout  -auto  multi_sequences.fa:seqID  |
  's/^>.\(*\)$/>newID \1/' > out.fa      출력물: >newID seqID:100-200
```

<서열ID> <start> <end> 형식으로 extract할 서열과 위치 정보를 수록한 텍스트 파일(regions)가 존재한다면 저장되어 있다면 다음과 같이 하면 된다. sed 명령어의 치환 부분에서 변수를 큰따옴표와 작은따옴표가 이중으로 둘러싸고 있음에 유의하라.

```
$ cat regions | while read id start end
> do
>  seqret  -stdout  -auto  -sbegin  $start  -send  $end  multi_sequences.fa:$id  |  sed
  's/^>.*$/>'"$id:$start-$end"'/' >> out.fa
> done
```

또 다른 방법으로는 samtools faidx 명령을 쓰는 것이다. 서열 추출 전에 fasta 파일에 대한 인덱스를 생성해 두어야 한다. EMBOSS의 seqret과 다른 점은 출력 서열의 ID는 seqID:start-end 형태로 새로 씌여진다는 것이다. samtools faidx로는 상보적인 서열로 전환이 되지 못하므로 필요시에는 EMBOSS 패키지의 revseq을 사용

한다. 필요하다면 파일로 리다이렉션하기 전에 위의 사례와 같이 sed 명령어를 사용하여 서열의 ID를 수정해도 된다.

```
$ samtools faidx multi_sequence.fa
$ samtools faidx multi_sequence.fa seqID 10-200 > out.fa
$ revseq out.fa out_rev.fa
```

▶ glimmer를 사용한 유전자 예측 결과물로부터 CDS 서열 추출하기

1. NCBI Glimmer site(http://www.ncbi.nlm.nih.gov/genomes/MICROBES/glimmer_3.cgi)에 세균 유전체 서열을 업로드하여 glimmer를 실행한 뒤 결과를 텍스트 파일로 저장하고 fasta 파일과 유사하게 정리한다. 입력하는 서열의 topology를 circular로 지정하면 서열의 말단을 통과하는 유전자 구조도 예측을 하게 되므로 주의를 요한다.
2. Glimmer는 각 (contig)서열에 대하여 동일한 orf 번호를 반복하여 사용하므로 입력물이 multi-fasta인 경우 혼동을 유발할 수 있다. 따라서 다음과 같은 awk 명령문을 이용하여 각 orf에 대한 유일한 식별자를 부여하도록 한다. 또한 glimmer가 출력하는 유전자의 시작 부위는 번역 개시위치 기준이다.

```
$ awk 'BEGIN {i=0} /^>/ {id=substr($0, 2)} /^orf/ {printf("%s %s ORF%05d %10s %d
 %d\n", id, $1, i, $2, $3, $4); i++}' glimmer_3.txt > glimmer_3.out
```

변환 전

orfID	start	end	frame	score
>contig_1				
orf00002	931	599	-2	11.47
orf00004	1710	1000	-1	8.84
orf00005	2312	1746	-3	9.51
orf00006	3031	2312	-2	10.54
orf00007	3256	3777	+1	8.57
>contig_2				
orf00001	1352	597	-3	3.39
orf00004	2598	2461	-1	3.54
orf00006	3654	4367	+3	5.46
orf00007	4858	4517	-2	3.12
>contig_3				
orf00002	597	1055	+3	2.24
orf00003	1033	1707	+1	3.12
orf00004	2657	2815	+2	7.00
orf00005	3488	3697	+2	2.52

변환 후

새로 생성

contig_1	orf00002	ORF00000	931	599	-2
contig_1	orf00004	ORF00001	1710	1000	-1
contig_1	orf00005	ORF00002	2312	1746	-3
contig_1	orf00006	ORF00003	3031	2312	-2
contig_1	orf00007	ORF00004	3256	3777	1
contig_2	orf00001	ORF00962	1352	597	-3
contig_2	orf00004	ORF00963	2598	2461	-1
contig_2	orf00006	ORF00964	3654	4367	3
contig_2	orf00007	ORF00965	4858	4517	-2
contig_3	orf00002	ORF01908	597	1055	3
contig_3	orf00003	ORF01909	1033	1707	1
contig_3	orf00004	ORF01910	2657	2815	2
contig_3	orf00005	ORF01911	3488	3697	2

3. 유전자 위치 정보를 암호화된 strand에 따라서 둘로 나누어 별도의 파일에 저장한다.

```
$ awk '$6 > 0' glimmer_3.out > glimmer_3_plus.out
$ awk '$6 < 0' glimmer_3.out > glimmer_3_minus.out
```

4. 다음을 실행하여 CDS 서열을 추출한다. plus 및 minus strand 각각을 실행할 때 입출력 파일을 바꾸는 것 말고도 어떤 점이 다른지를 유의하여 실행하라(파랑색으로 표시). 아미노산으로 번역하려면 EMBOSS의 transeq를 이용한다. glimmer가 출력하는 유전자 구조는 종결코돈까지를 표시하므로 번역을 마치면 마지막 아미노산 뒤에 '*'이 추가되며, 서열 ID 뒤에 '_1'이 부가된다. 이는 vi에서 수정하면 된다.

```
$ awk '{print $1, $3, $4, $5}' glimmer_3_plus.out | while read seqid ORFid start end
> do
```

```
> seqret -sbegin $start -send $end -stdout -auto contigs.fa:$seqid | sed
  's/^>.*$/'">$ORFid $seqid:$start-$end +"'/' >> orf_plus.fa
> done
$ awk '{print $1, $3, $5, $4}' glimmer_3_minus.out | while read seqid ORFid start
  end
> do
> seqret -sbegin $start -send $end -sreverse -stdout -auto contigs.fa:$seqid | sed
  's/^>.*$/'">$ORFid $seqid:$start-$end -"'/' >> orf_minus.fa
> done
$ cat orf_plus.fa orf_minus.fa > orf.fa
$ transeq orf.fa orf_translated.faa
```

5. seqret 대신 samtools faidx를 이용해서도 EMBOSS의 transeq를 이용하여 같은 작업을 할 수 있을 것이다.

▶ Multiple fasta 파일을 서열별로 분리하여 별도의 파일로 저장하기(fastaSplit.pl)

```
#!/bin/env perl
#
# Usage : fastaSplit.pl multiple_FASTA_file [output_directory]
#
use warnings;
use strict;

my $inputFile = shift;
my $outDir = shift || ".";
if ( ! -w $outDir ) {
   die "You cannot make files in this directory $outDir: $!";
}
open INPUT, $inputFile or die "Can't open $inputFile: $!";
while ( <INPUT> ) {
    if ( /^>([^\s^\t]+)/ ) {
        my $outFile = $1;
        my $outFileWithPath = "$outDir/$outFile";
        open OUT, ">$outFileWithPath";
        print STDERR "Writing file $outFileWithPath\n";
        print OUT $_;
    } else {
        print OUT $_;
    }
}
close OUT;
```

▶ Contig 서열 파일로부터 assembly 관련 수치 구하기(n50.pl)

```
#!/usr/bin/perl
#
# Source:
# http://genomics-array.blogspot.com/2011/02/calculating-n50-of-contig-assembly-file.html
#    Modified by Haeyoung Jeong
## Read Fasta File and compute length ###
#
my $length;
my $totalLength;
my @arr;
my $num;
```

```
while(<>){
   chomp;
   if(/^>([^\s]+)/){
       print " $length\n" unless $num == 0;
       $num++;
       print "SEQ: $1 ";
       push (@arr, $length);
       $totalLength += $length;
       $length=0;
  } else {
  s/\s//g;
  s/\t//g;
  $length += length($_);
  }
}
push (@arr, $length); # for the last contig
print " $length\n";
$totalLength += $length;

close(FH);

my @sort = sort {$b <=> $a} @arr;
my $n50;
print "$num contigs (total length: $totalLength)\n";
print "Max. contig length is ", $sort[0], "\n";
foreach my $val(@sort){
     $n50+=$val;
      if($n50 >= $totalLength/2){
         print "N50 is $val (N50 contig length is $n50)\n";
         last;
     }
}
print "Average contig length is ", $totalLength / $num, "\n";
```

참고 사이트

[Srrenu Vattipally] Bioinformatics I/O: Tips & tricks from a cluster of bioinformaticians
http://bioinformatics.cvr.ac.uk/blog/category/awk/

[Frederick J. Tan] sed, AWK, and bash scripting
https://emb.carnegiescience.edu/sites/default/files/140602-sedawkbash.key_.pdf

Bioinformatics one-liners
https://github.com/stephenturner/oneliners

seqtk: Toolkit for processing sequences in FAST/Q formats
https://github.com/lh3/seqtk

sed and awk for genomics
http://macmanes.com/sed-awk/