유전체 연구 실무진 역량 강화를 위한 리눅스 기초

홍창범 hongiiv@gmail.com

April 11, 2016

Contents

1 유전체 연구를 위한 리눅스

1.1 리눅스 시스템 정보 알아내기

본 문서는 리눅스 배포판[1]의 하나인 'Ubuntu (우분투)'를 기반으로 설명합니다. 별도의 표시가 없는 경우 모든 종류의 리눅스에 사용이 가능합니다. 리눅스는 다양한 배포판과 하드웨어상에서 동작하는 운영체제입니다. 자신의 리눅스가 어떠한 환경에서 동작하는지를 알아두어야 소프트웨어 설치시 자신의 리눅스에 적합한 소프트웨어의 설치가 가능합니다.

- 현재 자신이 사용하는 리눅스 배포판의 종류 식별하는 방법입니다. Ubuntu의 경우 무료로 배포되는 리눅스 운영체제로 현재 최신버전은 14.04 LTS (Long Term Support)[2] 버전입니다.

```
$ cat /etc/issue.net
Ubuntu 12.04.1 LTS
```

- 리눅스는 다양한 하드웨어 환경에서 운영되며 리눅스를 지원하는 소프트웨어들은 이러한 하드웨어에 따라 실행 파일을 따로 제공합니다. 따라서 현재 자신이 사용하는 하드웨어 정보를 알면 자신에게 맞는 소프트웨어를 다운로드하여 사용할 수 있습니다. 리눅스 서버 장비 하드웨어 사양 식별은 '-m' 즉, machine 옵션을 통해 알 수 있습니다. 'x86'은 Intel 기반의 CPU를 의미하며, '64'는 64비트 하드웨어를 의미[3]합니다.

```
$ uname -m
x86_64
```

- 커널은 리눅스 운영체제의 핵심으로 사용자의 명령을 실제 하드웨어를 통해 실행하도록 합니다. 리눅스 커널은 사용하는 배포판에 따라 서로 다른 버전을 사용합니다. 현재 가장 최신의 리눅스 커널은 3.14.3dmfh 2014년 5월6일 발표된 버전입니다. 각 리눅스 배포판은 이렇게 발표된 커널을 기반으로 제작됩니다. 리눅스의 커널 정보 식별 해보도록 하겠습니다.

```
$ uname -r
3.2.0-32-virtual
```

- 쉘은 리눅스 명령어를 입력받아 이를 실행하는 환경으로 'PATH'는 프로세스가 동작하는 방법에 영향을 끼치는 값인 환경 변수 중의 하나입니다. export는 이러한 환경변수의 값을 설정하는 명령어 입니다. 리눅스에 명령을 입력하면 PATH에 설정된 디렉토리를 우선 검색하여 해당 명령어가 있는지를 확인하고 이를 실행합니다. 따라서 자신의 직접 소프트웨어를 설치하고 리눅스 상에서 실행하는 경우 반드시 PATH를 지정해야 어디에서든지 실행이 가능하며 그렇지 않은 경우 소프트웨어가 설치된 디렉토리 내에서만 실행이 가능합니다. 쉘 환경 변수 확인은 'env' 명령으로 알아 낼 수 있으며, PATH는 'export'를 통해 설정합니다.

```
$ env | grep PATH
MANPATH=/usr/local/texlive/2013/texmf/doc/man:
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
INFOPATH=/usr/local/texlive/2013/texmf/doc/info:
$ export PATH=/BIO/app/bwa-0.7.5a/:$PATH
$ env | grep PATH
```

[1] 리눅스는 크게 레드햇 계열과 데비안 계열로 분리되며 각 계열별로 다양한 배포판이 존재한다.
[2] 코드명은 Trusty Tahr입니다.
[3] 흔히 줄여서 x64라고 표현합니다.

1.2 리눅스 파일 시스템

리눅스의 파티션은 하나의 물리적 디스크를 논리적으로 여러 영역으로 구분하여 관리하며 각 파티션은 파일 시스템을 생성하여 파일 및 디렉토리를 관리할 수 있습니다.

- 리눅스 시스템은 여러 사용자가 사용하는 시스템으로 각자 자신의 고유 영역인 홈디렉토리를 가지고 있습니다. 홈 디렉토리내에서는 자신이 파일을 생성, 삭제가 가능합니다. 홈 디렉토리로 이동하는 명령은 'cd' 명령이며, 현재 디렉토리 경로는 'pwd' 명령으로 확인할 수 있습니다.

```
$ cd
$ pwd
/home/hongiiv
```

- 디렉토리 만들고 해당 디렉토리로 이동하기

```
$ cd
$ mkdir sample_data
$ ls -la
total 2203488
drwxr-xr-x 16 hongiiv hongiiv       4096 May 29 10:34 .
drwxr-xr-x  3 root    root          4096 May  7 13:14 ..
-rw-------  1 hongiiv hongiiv       1908 May 10 11:59 .bash_history
-rw-r--r--  1 hongiiv hongiiv        220 May  7 13:14 .bash_logout
-rw-r--r--  1 hongiiv hongiiv       3763 May 10 17:06 .bashrc
drwxr-xr-x  2 root    root          4096 May 29 10:34 sample_data
$ cd sample_data
$ pwd
/home/hongiiv/sample_data
```

- 디렉토리 및 파일 삭제하기

```
$ cd
$ rm -rf sample_data
$ ls -la
total 2203488
drwxr-xr-x 16 hongiiv hongiiv       4096 May 29 10:34 .
drwxr-xr-x  3 root    root          4096 May  7 13:14 ..
-rw-------  1 hongiiv hongiiv       1908 May 10 11:59 .bash_history
-rw-r--r--  1 hongiiv hongiiv        220 May  7 13:14 .bash_logout
-rw-r--r--  1 hongiiv hongiiv       3763 May 10 17:06 .bashrc
$
```

- 리눅스 파일 시스템 보기

```
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/xvda1       19G   14G  4.8G  74% /
udev            3.9G  4.0K  3.9G   1% /dev
tmpfs           1.6G  188K  1.6G   1% /run
none            5.0M     0  5.0M   0% /run/lock
none            3.9G     0  3.9G   0% /run/shm
/dev/xvdb1       79G   38G   38G  50% /home/hongiiv/test
```

- 물리적 하드디스크 파티션 정보 보기 - 21.5 GB의 물리적인 /dev/xvda 하드디스크는 vxda1, xvda2 2개의 파티션으로 구성되어 있으며 각각 Linux, Linux swap의 파일시스템임을 확인할 수 있습니다.

```
$ fdisk -l
Disk /dev/xvda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders, total 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00034212

    Device Boot      Start         End      Blocks   Id  System
/dev/xvda1            2048    40038399    20018176   83  Linux
/dev/xvda2        40038400    41940991      951296   82  Linux swap / Solaris

Disk /dev/xvdb: 300.6 GB, 300647710720 bytes
171 heads, 35 sectors/track, 98112 cylinders, total 587202560 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x3459a991

    Device Boot      Start         End      Blocks   Id  System
/dev/xvdb1            2048   587202559   293600256   8e  Linux LVM
```

- 파일 시스템 마운트 정보 확인

```
$ cat /etc/fstab
proc /proc proc nodev,noexec,nosuid 0       0
/dev/xvda1 / ext3    errors=remount-ro 0       1
/dev/xvda2 none swap    sw              0       0
```

1.3 리눅스 하드디스크 추가하기

- fdisk를 통해 추가된 하드디스크를 확인한 후 파티셔닝, 파일시스템 생성, 마운트의 3단계를 거쳐 하드디스크를 사용합니다. USB 장치를 리눅스에 인식하기 위해서는 mount 과정만을 거치면 됩니다.

```
$ fdisk /dev/xvdb
$ mkfs.ext3 /dev/xvdb1
$ mkdir /new_hdd
$ mount /dev/xvdb1 /new_hdd
$ cd /new_hdd
$ df -h
```

1.4 파일 관련 명령어

- touch - 파일 크기가 0인 새로운 파일 생성하거나 파일이 생성된 시간을 변경할 수 있습니다. 간혹 유전체 관련 소프트웨어 설치나 교육시 사용하는 명령어로 숙지하시기 바랍니다.

```
$ touch a
$ ls -al
-rw-r--r--  1 root root      0 Jun 18 10:04 a
$ date
Wed Jun 18 10:05:10 KST 2014
```

```
$ touch -c a
$ ls -al
-rw-r--r--  1 root root      0 Jun 18 10:05 a
```

- cat - 파일의 내용을 확인하거나 간단한 스크립트 작성시 사용합니다. 'cat > test' 명령으로 test라는 파일을 생성하면서 파일 내용을 작성합니다. 작성이 완료된 후에는 'ctrl+D' 버튼을 눌러 빠져나올 수 있습니다.

```
$ cat > test
hi there
my name is hong
$ cat test
hi there
my name is hong
$ ls -al
-rw-r--r--  1 root root     25 Jun 18 10:09 test
```

- 특정 디렉토리의 파일의 갯수 세기

```
$ ls -l . | grep ^- | wc -l
50
```

- 파일의 특정 문자열로 시작하는 부분을 제외한 부분 출력하기입니다. VCF 파일과 같이 ''로 시작하는 부분은 주석인 경우 주석만을 제외한 실제 유전변이의 리스트를 출력합니다. 또는 그 반대로 주석 부분만을 출력합니다.

```
$ cd /DB/reference_data/gatk_bundle/2.8/hg19
$ grep -v "#" dbsnp_138.hg19.vcf| wc -l
$ grep -F "#" dbsnp_138.hg19.vcf |wc -l
165
```

- 특정 염색체만 출력합니다. 해당 염색체의 알파벳순 '-d', 숫자순'-c'으로 정렬이 가능합니다.

```
$ grep -v "#" dbsnp_138.hg19.vcf |awk '{print $1}'|more
chrM
chrM
chrM
chrM
chrM
chrM
chrM
chrM
chrM
$ grep -v "#" dbsnp_138.hg19.vcf |awk '{print $1}'| sort -d
chr1
chr2
$ grep -v "#" dbsnp_138.hg19.vcf |awk '{print $1}'| uniq -c
    475 chrM
4723878 chr1
4998710 chr2
4154091 chr3
$ grep -v "#" dbsnp_138.hg19.vcf | \
awk '{if ($1 == "chrM") printf "chrM␣is:␣%s\n", $2}'
chrM is: 16390
```

```
chrM is: 16391
chrM is: 16429
chrM is: 16445
chrM is: 16499
```

- 표준출력으로 출력되는 내용을 파일로 저장하기

```
$ grep -v "#" dbsnp_138.hg19.vcf | \
awk '{if ($1 == "chrM") printf "chrM␣is:␣%s\n", $2}' > chr_pos.txt
$ grep -v "#" dbsnp_138.hg19.vcf | \
awk '{if ($1 == "chr1") printf "chrM␣is:␣%s\n", $2}' >> chr_pos.txt
```

1.5 리눅스 네트워크 정보

- 네트워크 인터페이스에 대한 정보로 eth0의 inet addr이 외부에서 현재 리눅스로 접속 가능한 주소[4]입니다.

```
$ ifconfig
eth0      Link encap:Ethernet  HWaddr 02:00:5b:73:00:33
          inet addr:172.27.252.234  Bcast:172.27.255.255
          inet6 addr: fe80::5bff:fe73:33/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:501386 errors:0 dropped:0 overruns:0 frame:0
          TX packets:346879 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:19357734604 (1 GB)  TX bytes:2720265191 (2 GB)
          Interrupt:68

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:4337 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4337 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2203478 (2.2 MB)  TX bytes:2203478 (2.2 MB)
```

1.6 리눅스 압축 이해하기

리눅스는 다양한 압축을 지원하며, 소프트웨어나 데이터를 배포하는 경우 압축된 파일을 이용하여 배포합니다.

- 리눅스에서 사용하는 다양한 압축 해제 방법입니다.

```
$ cd compress
$ gzip -d compress01.gz
$ tar xvfz compress02.tar.gz
$ unzip compress03.zip
$ bzip -d comress04.bz2
$ tar xvfz compress05.tar.bz2
```

[4]리눅스 서버의 주소는 172.27.252.234로 각자의 실습 환경에 따라 다르게 표시된다.

- gzip: Recommended for fast network connections
- bzip2: Recommended for slower network connections (smaller size but takes longer to compress)
- zip: Not recommended but is provided as an option for those who cannot open the above formats
- 대용량의 압축된 유전체 데이터에 대해 압축을 해제하지 않고 미리 파일의 내용 확인하는 방법입니다. FASTQ 파일등을 확인하는데 유용합니다.

```
$ gzip -dc CEUTrio.HiSeq.WGS.b37.bestPractices.hg19.vcf.gz | more
$ gzip -dc CEUTrio.HiSeq.WGS.b37.bestPractices.hg19.tar.gz | tar -tvf -
```

1.7 리눅스 소프트웨어 설치하기

일반적으로 리눅스에 소프트웨어를 설치하는 방법은 다음의 3가지 방법이 있습니다. 첫번째는 바이너리 (실행) 파일을 압축형태로 제공하는 방법으로 간단히 압축을 해제하여 바로 사용이 가능하다. 두번째는 리눅스에서 제공하는 패키지를 이용하는 방법으로 우분투의 경우 APT라는 패키지 관리 프로그램을 이용한다. 세번째로는 소스파일을 이용하여 설치하는 방법이다.

1.7.1 APT를 이용한 소프트웨어 설치

- APT를 이용한 패키지 업데이트

```
$ apt-get update
$ apt-get install bwa
Reading package lists... Done
Building dependency tree
Reading state information... Done
Use 'apt-get autoremove' to remove them.
Suggested packages:
  samtools
The following NEW packages will be installed:
  bwa
0 upgraded, 1 newly installed, 0 to remove and 153 not upgraded.
Need to get 135 kB of archives.
After this operation, 286 kB of additional disk space will be used.
Fetched 135 kB in 3s (40.1 kB/s)
Selecting previously unselected package bwa.
(Reading database ...17 files and directories currently installed.)
Unpacking bwa (from .../archives/bwa_0.6.1-1_amd64.deb) ...
Processing triggers for man-db ...
Setting up bwa (0.6.1-1) ...
$ bwa

Program: bwa (alignment via Burrows-Wheeler transformation)
Version: 0.6.1-r104
Contact: Heng Li <lh3@sanger.ac.uk>

Usage:   bwa <command> [options]

Command: index         index sequences in the FASTA format
         aln           gapped/ungapped alignment
         samse         generate alignment (single ended)
         sampe         generate alignment (paired ended)
         bwasw         BWA-SW for long queries
         fastmap       identify super-maximal exact matches
```

```
        fa2pac          convert FASTA to PAC format
        pac2bwt         generate BWT from PAC
        pac2bwtgen      alternative algorithm for generating BWT
        bwtupdate       update .bwt to the new format
        bwt2sa          generate SA from BWT and Occ
        pac2cspac       convert PAC to color-space PAC
        stdsw           standard SW/NW alignment
```

- NGS 관련 소프트웨어 설치를 위해 미리 기본 설치되어야 하는 패키지 목록입니다.

```
$ apt-get update -y
$ apt-get install gcc -y
$ apt-get install make -y
$ apt-get install zlib1g-dev -y
$ apt-get install libncurses5-dev -y
$ apt-get install g++ -y
$ apt-get install tcl tk -y
$ apt-get install tcl-dev -y
$ apt-get install unzip -y
$ apt-get install screen -y
$ apt-get install python-dev -y
$ apt-get install python-software-properties -y
$ add-apt-repository ppa:webupd8team/java
$ apt-get update -y
$ apt-get install oracle-java7-installer -y
```

1.7.2 소스 코드 컴파일을 통한 소프트웨어 설치

- 소스로 설치하기

```
$ cd
$ cp /BIO/app/bwa-0.7.9a.tar.bz2 ./
$ tar xvfz bwa-0.7.9a.tar.bz2
$ cd bwa-0.7.9a
$ make
$ ./bwa

Program: bwa (alignment via Burrows-Wheeler transformation)
Version: 0.7.9a-r786
Contact: Heng Li <lh3@sanger.ac.uk>

Usage:   bwa <command> [options]

Command: index         index sequences in the FASTA format
         mem           BWA-MEM algorithm
         fastmap       identify super-maximal exact matches
         pemerge       merge overlapping paired ends (EXPERIMENTAL)
         aln           gapped/ungapped alignment
         samse         generate alignment (single ended)
         sampe         generate alignment (paired ended)
         bwasw         BWA-SW for long queries
```

```
        fa2pac          convert FASTA to PAC format
        pac2bwt         generate BWT from PAC
        pac2bwtgen      alternative algorithm for generating BWT
        bwtupdate       update .bwt to the new format
        bwt2sa          generate SA from BWT and Occ
```

2 Resequencing 데이터 분석하기

Sequencing 된 데이터를 Reference에 mapping한 후 유전변이를 찾아 이를 annotation하는 일련의 과정에 대해 설명합니다.

2.1 실습환경 설정하기

- 소프트웨어 PATH 설정하기

```
$ export PATH=/BIO/app/FastQC/:$PATH
$ export PATH=/BIO/app/bwa/:$PATH
$ export PATH=/BIO/app/samtools:$PATH
```

- Java Program 설치 위치

```
Picard: /BIO/app/picard/
GATK: /BIO/app/gatk/
```

- Reference Data 위치

```
BWA Rice6.1: bwa/reference/rice_6_1.fa
```

- 샘플 데이터 준비하기

```
$ cd
$ ln -s /BIO/data/bwa bwa
$ ls -R bwa
bwa:
fastq  reference

bwa/fastq:
rice_1.fastq  rice_2.fastq

bwa/reference:
illigal_chr  rice_6_1.fa.amb  rice_6_1.fa.ann
rice_6_1.fa.bwt  rice_6_1.fa.pac  rice_6_1.fa.sa
```

2.2 FastQC를 이용한 read data 통계 정보 얻기

- FastQC의 기본 옵션: '-t: 사용할 쓰레드의 갯수', '-o: 결과가 생성될 디렉토리를 지정[5]', '-f: 입력으로 사용되는 파일의 종류 fastq, bam, sam'
- 명령어 형식: **fastqc [옵션] Read파일명**
- FastQC 실행하기

```
$ cd
$ mkdir fastqc_out
$ fastqc -t 2 -o fastqc_out -f fastq bwa/fastq/rice_1.fastq
Started analysis of rice_1.fastq
Approx 5% complete for rice_1.fastq
Approx 10% complete for rice_1.fastq
```

[5]해당 디렉토리는 미리 만들어 두어야 합니다. (mkdir fastqc_out)

```
...
Approx 100% complete for rice_1.fastq
Analysis complete for rice_1.fastq
$ ls -al fastqc_out/
total 128
drwxr-xr-x 3 root    root       4096 Jun  2 11:23 .
drwxr-xr-x 8 hongiiv hongiiv    4096 Jun  2 11:23 ..
drwxr-xr-x 4 root    root       4096 Jun  2 11:23 rice_1_fastqc
-rw-r--r-- 1 root    root     112674 Jun  2 11:23 rice_1_fastqc.zip
```

- FastQC의 결과는 실행시 지정된 디렉토리에 생성되며 해당 디렉토리에는 입력파일의 이름으로 된 디렉토리에 HTML 형식의 결과가 생성됩니다.

2.3 BWA를 이용한 Reference 매핑

BWA를 이용한 reference 매핑은 paired-end read의 경우 'bwa aln'과 'bwa sampe'의 2단계를 거쳐서 진행되며 최종적으로 reference에 mapping된 정보가 기록된 txt 파일 포맷의 sam 파일이 생성됩니다.

- BWA aln의 기본 옵션: '-q', '-t'
- 명령어 형식: **bwa aln [옵션] reference fastq_파일**
- BWA aln 실행하기

```
$ mkdir bwa_out
$ bwa aln -q 20 -t 4 \
bwa/reference/rice_6_1.fa \
bwa/fastq/rice_1.fastq > bwa_out/rice_1.sai
$ bwa aln -q 20 -t 4 \
bwa/reference/rice_6_1.fa \
bwa/fastq/rice_2.fastq > bwa_out/rice_2.sai
$ ls -al bwa_out/rice_1.sai
-rw-r--r-- 1 root root 929816 Jun  2 11:47 bwa_out/rice_1.sai
$ ls -al bwa_out/rice_2.sai
-rw-r--r-- 1 root root 922208 Jun  2 11:49 bwa_out/rice_2.sai
```

- BWA sampe는 각각 reference에 매핑된 read의 pair 정보를 고려하여 매핑하는 과정입니다.
- 명령어 형식: **bwa sampe reference sai_1 sai_2 fastq_1 fastq_2**
- BWA sampe 실행하기

```
$ bwa sampe bwa/reference/rice_6_1.fa \
bwa_out/rice_1.sai bwa_out/rice_2.sai \
bwa/fastq/rice_1.fastq bwa/fastq/rice_2.fastq > bwa_out/rice.sam
$ ls -al bwa_out/rice.sam
$ tail bwa_out/rice.sam
```

- 최종 결과인 sam 파일은 txt 포맷으로 일반 문서 편집기 등을 이용하여 읽을 수 있습니다. tail 명령어는 파일의 끝 일부분을 보여주는 명령어로 해당 파일의 내용을 확인할 때 사용합니다. 이와 반대로 head 명령을 이용하여 파일의 앞부분 일부를 확인 할 수 있습니다.

2.4 Samtools를 이용한 BAM 파일 변환

Align된 sam 파일은 bam 포맷으로 변환 후 sorting 과정을 거쳐야 통계 및 variant calling의 작업을 수행할 수 있습니다. sam 파일을 bam 파일로 변환은 다양한 툴들이 지원하는 기능이며 여기서는 samtools를 이용하는 방법에 대해서 알아봅니다.

- SAM 파일을 BAM 파일로 변환 후 sorting 수행하기
- 명령어 형식: **samtools view -bSu sam_파일명 | samtools sort - bam_파일명**
- samtools view의 기본 옵션: '-b' 결과를 BAM 포맷으로 , '-S' input은 SAM 포맷, '-u' 압축되지 않은 BAM

```
$ samtools view -bSu bwa_out/rice.sam | \
samtools sort - bwa_out/rice.sorted
$ samtools index bwa_out/rice.sorted.bam
$ ls -al bwa_out/
total 23692
drwxr-xr-x 2 root    root        4 Jun  2 12:57 .
drwxr-xr-x 9 hongiiv hongiiv     4 Jun  2 11:46 ..
-rw-r--r-- 1 root    root      929 Jun  2 11:47 rice_1.sai
-rw-r--r-- 1 root    root      922 Jun  2 11:49 rice_2.sai
-rw-r--r-- 1 root    root    16930 Jun  2 11:53 rice.sam
-rw-r--r-- 1 root    root     4895 Jun  2 12:55 rice.sorted.bam
-rw-r--r-- 1 root    root      508 Jun  2 12:57 rice.sorted.bam.bai
```

2.5 Samtools를 이용한 BAM 파일 보기

- samtools view를 이용한 read 정보 보기

```
$ samtools view bwa_out/rice.sorted.bam chr3:10749666-10794394
```

- samtools tview를 이용한 read 정보 보기: 'g'를 눌러 'chr3:chr3:10749666'을 입력한 후 엔터키를 눌러 화살표키로 좌/우 이동합니다. 빠져나올 경우 'Esc'키를 눌러 종료합니다.

```
$ samtools tview bwa_out/rice.sorted.bam
```

2.6 Samtools를 이용한 BAM 통계 정보 얻기

- flagstat을 이용한 BAM 통계 정보

```
$ samtools flagstat bwa_out/rice.sorted.bam
50000 + 0 in total (QC-passed reads + QC-failed reads)
0 + 0 duplicates
49617 + 0 mapped (99.23%:-nan%)
50000 + 0 paired in sequencing
25000 + 0 read1
25000 + 0 read2
48498 + 0 properly paired (97.00%:-nan%)
49482 + 0 with itself and mate mapped
135 + 0 singletons (0.27%:-nan%)
212 + 0 with mate mapped to a different chr
66 + 0 with mate mapped to a different chr (mapQ>=5)
```

2.7 Picard를 이용한 BAM 파일 다루기

2.7.1 헤더 정보 변경하기

- Picard를 이용한 헤더 정보 변경하기

```
$ java -jar /BIO/app/picard/AddOrReplaceReadGroups.jar \
VALIDATION_STRINGENCY=LENIENT \
INPUT=bwa_out/rice.sorted.bam \
OUTPUT=bwa_out/rice.addgroup.sorted.bam \
RGID=test RGPL=illumina RGLB=test RGSM=test RGPU=manual
$ samtools view -H bwa_out/rice.addgroup.sorted.bam
 @HD   VN:1.4    SO:unsorted
 @SQ    SN:chr1 LN:43268879
 @SQ    SN:chr2 LN:35930381
 @SQ    SN:chr3 LN:36406689
 @SQ    SN:chr4 LN:35278225
 @SQ    SN:chr5 LN:29894789
 @SQ    SN:chr6 LN:31246789
 @SQ    SN:chr7 LN:29696629
 @SQ    SN:chr8 LN:28439308
 @SQ    SN:chr9 LN:23011239
 @SQ    SN:chr10        LN:23134759
 @SQ    SN:chr11        LN:28512666
 @SQ    SN:chr12        LN:27497214
 @RG    ID:test PL:illumina     PU:manual       LB:test SM:test
 @PG    ID:bwa  PN:bwa  VN:0.6.2-r126
```

2.7.2 Duplication Reads 제거하기

- Duplication Reads 제거하기

```
$ java -jar /BIO/app/picard/MarkDuplicates.jar \
I=bwa_out/rice.addgroup.sorted.bam \
REMOVE_DUPLICATES=false  \
VALIDATION_STRINGENCY=LENIENT \
AS=true \
OUTPUT=bwa_out/rice.mark.addgroup.sorted.bam \
METRICS_FILE=bwa_out/rice.mark.log
$ ls bwa_out/rice.mark.addgroup.sorted.bam
bwa_out/rice.mark.addgroup.sorted.bam
```

- Duplication 제거 후 BAM 통계 정보 비교하기

```
$ samtools flagstat bwa_out/rice.mark.addgroup.sorted.bam
50000 + 0 in total (QC-passed reads + QC-failed reads)
38 + 0 duplicates
49617 + 0 mapped (99.23%:-nan%)
50000 + 0 paired in sequencing
25000 + 0 read1
25000 + 0 read2
48498 + 0 properly paired (97.00%:-nan%)
49482 + 0 with itself and mate mapped
135 + 0 singletons (0.27%:-nan%)
```

```
212 + 0 with mate mapped to a different chr
66 + 0 with mate mapped to a different chr (mapQ>=5)
```

2.7.3 Picard를 이용한 Alignment 통계 구하기

- Alignment 통계 구하기

```
$ java -jar /BIO/app/picard/CollectAlignmentSummaryMetrics.jar \
I=bwa_out/rice.mark.addgroup.sorted.bam \
OUTPUT=bwa_out/rice_bam_summery.txt
$ head bwa_out/rice_bam_summery.txt
...
CATEGORY         TOTAL_READS     PF_READS         PCT_PF_READS
FIRST_OF_PAIR    25000    25000    1
SECOND_OF_PAIR   25000    25000    1
PAIR     50000    50000    1
```

- insert size 통계 구하기

```
$ java -jar /BIO/app/picard/CollectInsertSizeMetrics.jar \
HISTOGRAM_FILE=bwa_out/insert_rice_hist.png \
INPUT=bwa_out/rice.mark.addgroup.sorted.bam  \
OUTPUT=bwa_out/insert_rice.txt
$ ls -al bwa_out | grep insert
-rw-r--r-- 1 root     root    7984 Jun  2 17:02 insert_rice_hist.png
-rw-r--r-- 1 root     root    4007 Jun  2 17:02 insert_rice.txt
```

2.7.4 Mate 정보 fix하기

- FixMateInformation을 이용한 Mate 정보 fix하기

```
$ java -jar /BIO/app/picard/FixMateInformation.jar \
I=bwa_out/rice.mark.addgroup.sorted.bam \
SORT_ORDER=coordinate \
VALIDATION_STRINGENCY=LENIENT \
CREATE_INDEX=true \
O=bwa_out/rice_final.bam
$  ls -al bwa_out/rice_final.*
-rw-r--r-- 1 root root  508968 Jun  2 14:31 bwa_out/rice_final.bai
-rw-r--r-- 1 root root 5009071 Jun  2 14:31 bwa_out/rice_final.bam
```

2.8 GATK UnifiedGenotyper를 이용한 Variant Calling

- Variant Calling

```
$ java -jar /BIO/app/gatk/GenomeAnalysisTK.jar \
-T UnifiedGenotyper \
-glm SNP \
-R /DB/galaxyIndices/genomes/Osjaponica/rice_6_1/seq/rice_6_1.fa \
-stand_call_conf 50.0 \
-stand_emit_conf 10.0 \
```

```
-dcov 200 \
-o bwa_out/rice.snp.raw.vcf  \
-I bwa_out/rice_final.bam
$ tail bwa_out/rice.snp.raw.vcf
$ grep -F "#" bwa_out/rice.snp.raw.vcf |wc -l
39
$ grep -Ev "#" bwa_out/rice.snp.raw.vcf|wc -l
540
```

2.9 Variant Annotation

- SnpEff를 이용한 Annotation

```
$ java -jar /BIO/app/snpEff_3_1/snpEff.jar \
eff -q -c /BIO/app/snpEff_3_1/snpEff.config \
-i vcf  \
rice6.1 \
bwa_out/rice.snp.raw.vcf \
-stats bwa_out/rice.snpeff.stat > bwa_out/rice.snpeff.vcf
$ ls bwa_out | grep rice.snpeff
rice.snpeff.stat
rice.snpeff.stat.genes.txt
rice.snpeff.vcf
```

2.10 Resequencing 데이터 압축 및 다운로드

- 분석 결과 파일 및 디렉토리 목록 확인하기

```
$ ls -al bwa_out
total 48684
drwxr-xr-x 2         4096 Jun  2 16:30 .
drwxr-xr-x 9         4096 Jun  2 16:32 ..
-rw-r--r-- 1       929816 Jun  2 11:47 rice_1.sai
-rw-r--r-- 1       922208 Jun  2 11:49 rice_2.sai
-rw-r--r-- 1      4991824 Jun  2 13:23 rice.addgroup.sorted.bam
-rw-r--r-- 1       508968 Jun  2 14:31 rice_final.bai
-rw-r--r-- 1      5009071 Jun  2 14:31 rice_final.bam
-rw-r--r-- 1      4991958 Jun  2 14:25 rice.mark.addgroup.sorted.bam
-rw-r--r-- 1         2479 Jun  2 14:25 rice.mark.log
-rw-r--r-- 1     16930320 Jun  2 11:53 rice.sam
-rw-r--r-- 1       328516 Jun  2 16:20 rice.snpeff.stat
-rw-r--r-- 1      9325321 Jun  2 16:20 rice.snpeff.stat.genes.txt
-rw-r--r-- 1       243293 Jun  2 16:20 rice.snpeff.vcf
-rw-r--r-- 1        99266 Jun  2 14:54 rice.snp.raw.vcf
-rw-r--r-- 1         3717 Jun  2 14:54 rice.snp.raw.vcf.idx
-rw-r--r-- 1      4895675 Jun  2 12:55 rice.sorted.bam
-rw-r--r-- 1       508968 Jun  2 12:57 rice.sorted.bam.bai
```

- 결과 데이터 압축하기

```
$ tar cvfz bwa_out.tar.gz ./bwa_out/
./bwa_out/
```

```
./bwa_out/rice.snpeff.stat
./bwa_out/rice.snpeff.vcf
./bwa_out/rice.mark.addgroup.sorted.bam
./bwa_out/rice.snpeff.stat.genes.txt
./bwa_out/rice_2.sai
./bwa_out/rice.snp.raw.vcf
./bwa_out/rice_final.bai
./bwa_out/rice_1.sai
./bwa_out/rice.snp.raw.vcf.idx
./bwa_out/rice.sam
./bwa_out/rice.addgroup.sorted.bam
./bwa_out/rice.sorted.bam
./bwa_out/rice.sorted.bam.bai
./bwa_out/rice.mark.log
./bwa_out/rice_final.bam
$ ls -al bwa_out.tar.gz
-rw-r--r-- 1 root root 26687772 Jun  2 16:32 bwa_out.tar.gz
```

- FTP 클라이언트 프로그램[6]을 실행[**?**]하여 리눅스 서버[7]로 접속[**?**] 합니다. 'bwa_out.tar.gz' 파일을 선택하여 다운로드합니다.

[6] https://www.altools.co.kr/EstLab/ALFTPServer.aspx
[7] 1.5 리눅스 네트워크를 참고하여 주소를 넣습니다.

3 RNA-Seq 데이터 분석하기

3.1 샘플 데이터 준비하기

- 샘플 데이터 준비하기

```
$ cd
$ ln -s /BIO/data/tophat tophat
$ cd tophat
$ ls -la
total 20
drwxr-xr-x 5 root root 4096 May 29 14:54 .
drwxr-xr-x 6 root root 4096 May 29 14:54 ..
drwxr-xr-x 2 root root 4096 May 29 14:50 fastq
drwxr-xr-x 2 root root 4096 May 29 14:47 gtf
drwxr-xr-x 2 root root 4096 May 29 14:47 reference
```

3.2 Align the RNA-seq reads to the genome

- Map the reads for each sample to the reference genome (분석 소요시간 각 6분)

```
$ cd
$ export PATH=/BIO/app/tophat-2.0.6.Linux_x86_64/:$PATH
$ export PATH=/BIO/app/bowtie2-2.0.3:$PATH
$ export PATH=/BIO/app/samtools-0.1.18:$PATH
$ mkdir tophat_out
$ tophat -p 4 -G tophat/gtf/genes.gtf \
 -o tophat_out/C1_R1_thout \
 tophat/reference/dm3 \
 tophat/fastq/C1_R1_1.fastq tophat/fastq/C1_R1_2.fastq
$ tophat -p 4 -G tophat/gtf/genes.gtf \
 -o tophat_out/C2_R1_thout \
 tophat/reference/dm3 \
 tophat/fastq/C2_R1_1.fastq tophat/fastq/C2_R1_2.fastq
```

- Tophat 결과 파일 및 디렉토리[8]

```
$ ls -R tophat_out
```

3.3 Assemble expressed genes and transcripts

- Assemble expressed genes and transcripts (소요시간 1분)

```
$ export PATH=/BIO/app/cufflinks-2.0.2.Linux_x86_64:$PATH
$ cufflinks -p 4 -o tophat_out/C1_R1_thout \
tophat_out/C1_R1_thout/accepted_hits.bam
$ cufflinks -p 4 -o tophat_out/C2_R1_thout \
tophat_out/C2_R1_thout/accepted_hits.bam
```

- Cufflinks 결과 gtf 파일 확인하기

[8] 결과 디렉토리의 accepted_hits.bam 파일을 각각 확인하세요.

```
$ ls -al tophat_out/C1_R1_thout/transcripts.gtf
$ ls -al tophat_out/C2_R1_thout/transcripts.gtf
```

- Create a file called assemblies.txt that list the assembly file for each sample.

```
$ echo 'tophat_out/C1_R1_thout/transcripts.gtf' > \
tophat_out/assemblies.txt
$ echo 'tophat_out/C2_R1_thout/transcripts.gtf' >> \
tophat_out/assemblies.txt
$ cat tophat_out/assemblies.txt
tophat_out/C1_R1_thout/transcripts.gtf
tophat_out/C2_R1_thout/transcripts.gtf
```

3.4 Create a single merged transcriptome annotation

- 각 샘플의 transcriptome annotation 합치기 (소요시간 1분)

```
$ cuffmerge -o tophat_out/merged_asm \
-g tophat/gtf/genes.gtf  \
-s tophat/reference/dm3.fa \
-p 4 \
tophat_out/assemblies.txt
$ ls -al tophat_out/merged_asm/
```

3.5 Identify differentially expressd genes and transcripts

- 두샘플간의 gene 및 transcripts의 차이 찾기[9]

```
$ cuffdiff -o tophat_out/diff_out -b tophat/reference/dm3.fa \
-p 4 -L C1,C2 \
-u tophat_out/merged_asm/merged.gtf \
tophat_out/C1_R1_thout/accepted_hits.bam \
tophat_out/C2_R1_thout/accepted_hits.bam
```

- cuffdiff 결과 확인하기

```
$ ls -al tophat_out/diff_out/
total 92
.
..
cds.count_tracking
cds.diff
cds_exp.diff
cds.fpkm_tracking
cds.read_group_tracking
gene_exp.diff
genes.count_tracking
genes.fpkm_tracking
genes.read_group_tracking
isoform_exp.diff
```

[9] -L 옵션의 C1과 C2사이의 콤마(,) 양쪽에는 공백이 없어야 합니다.

```
isoforms.count_tracking
isoforms.fpkm_tracking
isoforms.read_group_tracking
promoters.diff
read_groups.info
run.info
splicing.diff
tss_group_exp.diff
tss_groups.count_tracking
tss_groups.fpkm_tracking
tss_groups.read_group_tracking
```

3.6 CummeRbund 패키지를 이용한 differential 분석 결과 다루기

3.6.1 각 샘플의 expression level의 분포 plot 그리기

- 스크립트 내용 확인하기

```
$ more tophat/scripts/plot01.R
library(cummeRbund)
cuff_data<-readCufflinks('tophat_out/diff_out')
png(filename = "tophat_out/plot01.png", width = 1280, height = 960)
tryCatch({
  csDensity(genes(cuff_data))
},error = function(e) {paste("expressionbarplot failed: ", e)})
devname = dev.off()
$
```

- Plot 그리기

```
$ R CMD BATCH tophat/scripts/plot01.R
$ ls -al tophat_out/
total 88
drwxr-xr-x 6 root root  4096 May 29 17:07 .
drwxr-xr-x 7 root root  4096 May 29 17:07 ..
-rw-r--r-- 1 root root    78 May 29 17:02 assemblies.txt
drwxr-xr-x 3 root root  4096 May 29 17:02 C1_R1_thout
drwxr-xr-x 3 root root  4096 May 29 17:02 C2_R1_thout
drwxr-xr-x 2 root root  4096 May 29 17:06 diff_out
drwxr-xr-x 3 root root  4096 May 29 17:02 merged_asm
-rw-r--r-- 1 root root 55882 May 29 17:07 plot01.png
$
```

3.6.2 두 샘플의 gene expression 비교 plot 그리기 (scatter plot)

3.6.3 두 샘플의 gene expression 비교 plot 그리기 (volcano plot)

3.6.4 특정 유전자의 expression level 비교 (bar plot)

3.6.5 두 샘플간의 유의한 gene expression 목록 출력하기

- 두 샘플간 유의한 차이를 보이는 유전자 목록 출력 스크립트

```
$ more tophat/scripts/diff_gene.R
library (cummeRbund)
cuff_data < - readCufflinks ('tophat_out / diff_out')
gene_diff_data <- diffData(genes(cuff_data))
sig_gene_data <- subset(gene_diff_data, (significant == 'yes'))
write.table(sig_gene_data, 'diff_genes.txt',
  sep='\t',row.names = F, col.names = T, quote = F)
write.table(sig_gene_data, 'tophat_out/diff_genes.txt',
  sep='\t',row.names = F, col.names = T, quote = F)
```

- 두 샘플간 유의한 차이를 보이는 유전자 목록 출력하기

```
$ R CMD BATCH tophat/scripts/diff_gene.R
$ more tophat_out/diff_genes.txt
gene_id  sample_1       sample_2        status  value_1 significant
XLOC_000002     C1      C2      OK      11425.8 ...     yes
XLOC_000003     C1      C2      OK      428911  ...     yes
XLOC_000006     C1      C2      OK      180668  ...     yes
XLOC_000008     C1      C2      OK      887.207 ...     yes
$
```

3.7 RNA-Seq 결과 데이터 압축 및 다운로드

지금까지 자신의 홈디렉토리의 tophat_out에 모든 분석 결과가 존재합니다. 이제 이 디렉토리를 압축하여 자신의 로컬 PC로 복사합니다.

- 분석 결과 파일 및 디렉토리 목록 확인하기

```
$ ls -al tophat_out/
total 88
drwxr-xr-x 6 root root  4096 May 29 17:07 .
drwxr-xr-x 7 root root  4096 May 29 17:07 ..
-rw-r--r-- 1 root root    78 May 29 17:02 assemblies.txt
drwxr-xr-x 3 root root  4096 May 29 17:02 C1_R1_thout
drwxr-xr-x 3 root root  4096 May 29 17:02 C2_R1_thout
drwxr-xr-x 2 root root  4096 May 29 17:06 diff_out
drwxr-xr-x 3 root root  4096 May 29 17:02 merged_asm
-rw-r--r-- 1 root root 55882 May 29 17:07 plot01.png
```

- 결과 데이터 압축하기

```
$ cd
$ tar cvfz rna_seq_results.tar.gz ./tophat_out
./tophat_out/
./tophat_out/tmp/
./tophat_out/prep_reads.info
./tophat_out/logs/
./tophat_out/logs/tophat.log
./tophat_out/logs/run.log
$ ls -al
rna_seq_results.tar.gz
```

- FTP 클라이언트 프로그램[10]을 실행[**?**]하여 리눅스 서버[11]로 접속[**?**] 합니다. 'rna_seq_result.tar.gz' 파일을 선택하여 다운로드합니다.

[10] https://www.altools.co.kr/EstLab/ALFTPServer.aspx
[11] 1.5 리눅스 네트워크를 참고하여 주소를 넣습니다.

References