### 201801 case (100개 샘플이 존재한다고 가정) ###

Step-1

### bininfo 생성 ###

# cd /BiO/Alba/Enet/201801/

# qsub -cwd -S /usr/bin/python -l hostname=alba /BiO/Alba/Enet/code/copy\_fastq.py 201801\_match\_fastq.list

# ls \*.Fastq > flist

# sed -n '1,50p' flist > flist1; sed -n '51,100p' flist > flist2

# qsub -cwd -S /bin/sh -l hostname=alba /BiO/Alba/Enet/code/enet\_analysis.sh flist1

# qsub -cwd -S /bin/sh -l hostname=alba /BiO/Alba/Enet/code/enet\_analysis.sh flist2

# ls \*.Fastq.sam.bam.sort.bam.rmdup.bam.sam > blist

# qsub -cwd -S /bin/sh -l hostname=alba /BiO/Alba/Enet/code/bininfo.sh blist

**### 최종 bininfo list check ###**

**## N차에서 생성하고자하는 bininfo 작업이 모두 끝난 후에 하시면 됩니다.**

# ls /BiO/Alba/2017\_backup/SeqFF/ENET/bininfo/test\_data/\*\_bininfo > infolist

## N차에서 생성한 샘플에 대한 모든 정보가 info.N.list에 포함되어야합니다. (5st : 201601, 201602, 201610, 201801, 201802 모두 포함)

# ex) cut -f1-5 /BiO/Alba/Enet/201801/201801\_match\_fastq.list >> /BiO/Alba/Enet/analysis/info.5st.list

# ex) cut -f1-5 /BiO/Alba/Enet/201802/201802\_match\_fastq.list >> /BiO/Alba/Enet/analysis/info.5st.list

# cp /BiO/Alba/2017\_backup/SeqFF/ENET/bininfo/test\_data/infolist /BiO/Alba/Enet/analysis/

# cd /BiO/Alba/Enet/analysis/

# ./match\_2.py info.5st.list infolist

## infolist의 모든 샘플이 **final.list에 존재하는지 check. 개수가 맞지 않으면 except.list를 확인해주세요.**

# mv final.list final.5st.list

# mv /BiO/Alba/2017\_backup/SeqFF/ENET/bininfo/test\_data/\*\_bininfo /BiO/Alba/2017\_backup/SeqFF/ENET/bininfo/

### Training ###

## input-data/ 폴더에 bininfo\_5000 옮기기

## /BiO/Alba/Enet/Training/code/Enet.py의 input file을 bininfo\_5000으로 변경

# cd /BiO/Alba/Enet/Training/fold\_result/

# rm \*

# qsub -cwd -S /usr/bin/python -l hostname=alba /BiO/Alba/Enet/Training/code/Enet.py

|  |  |
| --- | --- |
| ls –alh \* | 파일 자세히 보기 |
| chmod u+x 파일이름.py | 실행파일로 변경(Python 코드 맨 윗줄에 #!/usr/bin/python 추가 해야 함) |
| vi /proc/cpuinfo | cpu 개수 확인 |
| ssh –p 3030 lion / ssh –p 3030 tiger | lion/tiger server 접속 |
| qstat | 작업 모니터링(자매품~ qstat –g c, qstat –f) |
| qdel PID | 작업 삭제 |
| ps -lf  ps -aux | 프로세스 상태 확인 |
| qsub –cwd –S /bin/sh –l hostname=genomecare –pe smp 5 test.sh | -S : script  -l : hostname  -pe smp : 사용할 cpu 개수 |
| df –h | storage 용량 확인 |
| wget download\_url | file download |
| apt-get install 설치할거 | install |
| nohup 명령어 & | & : background 에서 돌리기  nohub : 로그아웃 되도 server에서 계속 실행 |
| jobs | background에서 돌아가는 프로세스 확인 |
| rm -r dir/ | 폴더 강제 삭제(\*\*조심) |
| cat plink.lmiss | sort -k 5 -r -n >> plink.sort.lmiss | -k : 열 선택 -r : 역순 -n : 숫자만 포함 |

**Linux 명령어 – cut/sed/paste**

열을 잘라서 출력하고 싶은 경우 = cut

cut –f8 filename > newfilename : filenam의 8열을 잘라서 newfilename으로 생성.

cut –f3-4 filename > newfilename : 3, 4열을 잘라서 생성.

행을 잘라서 출력하고 싶은 경우 = sed

sed –n ‘1,3p’ filename > newfilename : 1~3 행을 출력.

sed '1,26d' filename > newfilename : 1~26 행을 삭제하고 출력

파일을 합치고 싶은 경우 = paste

paste file1 file2 file3 –d’\t’ > newfilename : 합치고 싶은 파일 인 file1, file2, file3를 입력 후 –d 를 이용해 tab을 중간에 넣어 준다

|  |  |
| --- | --- |
| kill PID | process kill |
| rename 's/\.ss$/\.ress/' \*.ss | renaming |
| cat filename | wc -l | file line 수 계산 |
| tar -zxvf file.tar.gz | tar file 압축해제 |
| touch filename | file 생성 |
| cat file\* >> allfile | >> : file\* 내용을 allfile에 이어서 쓰기( > 는 덮어쓰기) |
| chmod 664 file | rwx : 421 |
| sed 's/......$//' delete1.list > delete.list | 뒤에서 6자리 자르고 출력 |
| %s/^M//g | ^M 제거(^M은 반드시 ctrl+V, ctrl+M으로 입력) |
| find ./ -name "\*.html" -exec sed -i 's/old/new/g' {} \; | 특정 문자열 치환 명령어 |
| ctrl + z | 프로세스 작업 중단(Stopped) |
| bg %N | 프로세스 백그라운드 전환 |
| disown | 소유권 해제 -> nohup한 것과 같음 |
| ln -s 대상경로 링크경로 | symbolic link 생성 |

**Linux scp를 위한 공개키 등록**

1. **ssh-keygen -t rsa**

~/.ssh/id\_rsa.pub 파일의 존재 여부 check. 없다면 위 명령어를 이용하여 public key, private key를 생성

RSA 암호화 방식

1. **ssh-copy-id -I ~./ssh/id\_rsa.pub [user]@[host**]

생성한 공개키를 접속하고자 하는 원격 서버에 등록해준다. ssh-copy-id 명령어를 사용하면 원격 서버의 authorized\_keys 파일에 공개키를 추가해준다.

원격서버에 공개키가 추가되면 scp를 사용하여 서버간 파일 복사를 할 때, 인증 절차를 생략할 수 있다.

현재 Backup server 에 albatross의 공개키가 등록되어 있음 -> scp를 통해 backup서버의 파일을 복사해 오는 과정에서 albatross의 비밀번호를 입력하는 과정을 생략한다

**Ndata\_analysis\_all.sh**

|  |  |
| --- | --- |
| for i in `ls \*fq`;do `**cat $i|sed 's/\(.\{35\}\).\*/\1/'>$i.fastq**`;done | 모든 fq 파일의 read를 숫자 크기로 잘라내어 .fastq로 저장한다. |
| for fq in `ls \*.Fastq`;do ` **bowtie2 -p 12 -x /BiO/NIPT/UCSC\_hg19/index $fq -S $fq.sam**`;done | .Fastq 파일을 정렬시켜서 .sam 파일로 만든다. –p는 병렬 스레드 설정. reference genome은 hg19를 사용한다. |
| for aa in `ls \*.sam`;do `**wc -l $aa >$aa.rawread**`;done | .sam 파일을 이용해서 read 수를 계산한다. 각 chromosome 별 read 수를 계산하는 파일은 chr\_count.sh |
| for aa in `ls \*.sam`;do `**samtools view -Sb $aa >$aa.bam**`;done | .sam 파일을 binary 형식인 bam 파일로 저장한다. bam 파일 사용시 속도 향상. |
| for aa in `ls \*.bam`;do `**samtools sort $aa -o $aa.sort.bam**`;done | .bam 파일을 정렬한다. |
| for aa in `ls \*sort.bam`;do `**samtools index $aa**`;done | .bam 파일을 인덱싱 한다. |
| for aa in `ls \*sort.bam`;do `**samtools rmdup -s $aa $aa.rmdup.bam**`;done | .bam 파일을 이용해서 duplication을 제거한다. |
| for aa in `ls \*rmdup.bam`;do `**samtools view $aa >$aa.sam**`;done | .bam 파일을 다시 .sam 파일로 만든다. |
| for un in `ls \*rmdup.bam.sam`;do `**cat $un|grep -v 'XS:i'|awk '$5==42 {print $0}'|grep 'AS:i:0**'>$un.unique`;done | unique read만 뽑아서 저장한다. 앞 슬라이드에 설명. |
| ./chr\_count.sh | 각 chromosome 별 read 수를 계산. |

**Bowtie2, samtools 명령어**

\* bowtie2 option : -p(병렬 스레드 설정), -x(reference genome 설정), -S(SAM file)

**1.정렬**

bowtie2 -p 12 -x /BiO/NIPT/UCSC\_hg19/index name.Fastq -S name.Fastq.sam

**2.bam 파일로 format 변경**

samtools view -Sb name.Fastq.sam > name.Fastq.sam.bam

**3.Sort**

samtools sort name.Fastq.sam.bam -o name.Fastq.sam.bam.sort.bam

**4.Index**

samtools index name.Fastq.sam.bam.sort.bam

**5.Duplication remove**

samtools rmdup -s name.Fastq.sam.bam.sort.bam name.Fastq.sam.bam.sort.bam.rmdup.bam

**Bowtie2, samtools 명령어**

1. **wc -l a.Fastq.sam > a.Fastq.sam.rawread : read 수 계산하는 명령어.**

**2.cat rmdup.bam.sam | grep –v ‘XS:i’ | awk ‘$5==42 {print $0}’ | grep ‘AS:i:0’ > rmdup.bam.sam.unique**

🡪 grep –v ‘XS:i’ : XS:i가 없는 line 만 출력. 이때 XS:i는 read에 대해 2개 이상의 정렬이 발견되는 경우이다.

* awk ‘$5==42 {print $0}’ : 5번째 필드가 42인 경우에만 모든 라인을 출력한다. 이때 5번째 필드가 42인 것은 bowtie2에서 MAPQ의 최대값이 42이기 때문이다.
* Grep ‘AS:i:0’ : AS:i:0인 라인만 출력. AS:i:0는 정렬점수.\

**samtools vcf(snp call)**

1. **Pileup file 생성**

samtools mpileup -uf hg19.fa a.sort.bam -o test.bcf

samtools mpileup a.sort.bam > a.sort.bam.pileup

1. **Bcftools, htslib download**

git clone https://github.com/samtools/bcftools.git

git clone https://github.com/samtools/htslib.git

1. **다운 폴더로 이동 및 install**

cd bcftools

make

**4.경로 설정**

export PATH=경로:$PATH

**5.SNP call**

bcftools call -mv -Ov test.bcf -o test.snpcall.vcf

**6.INDEL 제거**

bcftools view --exclude-types indels test.snpcall.vcf –o test.snpcall.dell.vcf

**imputation final command**

**# phasing**

**1)gotcloud-gotcloud.1.17/src/vcfCooker/bin/vcfCooker** --in-vcf ../Genome\_care\_chX\_redu.vcf.gz --write-bed --out Genome\_care\_chX\_redu

2./BiO/NIPT/Analysis/minjung/imputation/shapeit\_static/bin/shapeit --input-bed Genome\_care\_chX\_redu.bed Genome\_care\_chX\_redu.bim Genome\_care\_chX\_redu.fam --input-map genetic\_map\_b37/genetic\_map\_chrX\_combined\_b37.txt --output-max Genome\_care\_chX\_redu.phased --effective-size 11418

3./BiO/NIPT/Analysis/minjung/imputation/shapeit\_static/bin/shapeit -convert --input-haps Genome\_care\_chX\_redu.phased --output-vcf Genome\_care\_chX\_redu.phased.vcf

**# imputation**

1. nohup /BiO/NIPT/Analysis/minjung/imputation/Minimac3/bin/Minimac3 --refHaps /BiO/NIPT/Analysis/minjung/imputation/shapeit\_static/example/1000genome/ALL.chr12.phase3\_v5.shapeit2\_mvncall\_integrated.noSingleton.genotypes.vcf --haps /BiO2/NIPT/Analysis/analysis\_genomom/GWAS95/PLINK/chr/phase/Genome\_care\_ch1\_redu.phased.vcf --prefix impu\_phased\_chr1 --chr 1 --noPhoneHome --format GT,DS,GP --allTypedSites --cpus 30 &

**# filter & ped**

/BiO/NIPT/Analysis/minjung/ngs/bcftools/bcftools view --exclude-types indels impu\_phased\_chr6.dose.vcf -o impu\_phased\_chr6.dose.filter.vcf

vcftools --vcf ../impu\_phased\_chr9.dose.filter.vcf --out impu\_phased\_chr9 --plink

plink --file impu\_phased\_chr9 --geno 0.1 --mind 0.1 --hwe 1e-3 --recode --out impu\_phased\_chr9\_filter --noweb

**# reformat & rename**

./reformat.py impu\_phased\_chr9\_filter.ped

./rename.py re\_impu\_phased\_chr9\_filter.ped

mv rename\_re\_impu\_phased\_chr9\_filter.ped final\_impu\_chr9.ped

mv impu\_phased\_chr9\_filter.map final\_impu\_chr9.map

**# location of final ped/map file**

/BiO2/NIPT/Analysis/analysis\_genomom/GWAS95/PLINK/re\_imputation/ped/final

|  |
| --- |
| **mininac3 imputation 명령어** |
| /BiO/NIPT/Analysis/minjung/imputation/Minimac3/bin/Minimac3 --refHaps ../../shapeit\_static/example/1000genome/ALL.chr6.phase3\_v5.shapeit2\_mvncall\_integrated.noSingleton.genotypes.vcf --haps targetStudy.vcf --prefix test1000genome --chr 1 --noPhoneHome --format GT,DS,GP --allTypedSites --cpus 60 |
| --refHaps reference panel --noPhoneHome : Michigan serve로 전달 x  --haps vcf file --format <GT|DS|GP>  --cpus cpu 개수 --allTypedSites |
| **plink 명령어** |
| /BiO/NIPT/Analysis/minjung/plink/plink --file Genome\_Care --recode vcf --no-pheno --no-fid --no-parents --no-sex --out Genome\_Care |
| --file map/ped file  --recode vcf : vcf 로 변환  --no-pheno --no-fid --no-parents --no-sex : phenotype, familyID, parents, sex 없음 |
| /BiO/NIPT/Analysis/minjung/plink/plink --vcf <input\_vcf\_name> --recode ped --out <output\_plink\_name> |
| .vcf file을 plink forma으로 convert. |
| plink --bfile Genome\_care\_ch3\_redu --missing --noweb  (cat plink.lmiss | sort -k 5 -r -n >> plink.sort.lmiss) |
| .bed, .bim, .fam file의 missing SNP log create.  -- noweb : web 연결 안함 |
| **vcfCooker** |
| gotcloud-gotcloud.1.17/src/vcfCooker/bin/vcfCooker --in-vcf ch5\_redu.vcf.gz --write-bed --out ch5\_redu |
| .vcf file convert to binary format(.bed & .bim & .fam) |

|  |
| --- |
| **vcftools 명령어** |
| vcftools --vcf imputation\_chr1.vcf --out imputation\_chr1 --plink |
| vcf file을 plink format(.ped .map)으로 convert. |
| **shapeit phase 명령어** |
| /BiO/NIPT/Analysis/minjung/imputation/shapeit\_static/bin/shapeit -V 1000genome/ALL.chr1.phase3\_v5.shapeit2\_mvncall\_integrated.noSingleton.genotypes.vcf -M genetic\_map\_chr1\_combined\_b37.txt -O 1000genomechr1.phased |
| --V vcffile  --M map file  -O output file |
| /BiO/NIPT/Analysis/minjung/imputation/shapeit\_static/bin/shapeit --input-bed Genome\_care\_ch5\_redu.bed Genome\_care\_ch5\_redu.bim Genome\_care\_ch5\_redu.fam --input-map genetic\_map\_b37/genetic\_map\_chr5\_combined\_b37.txt  --output-max Genome\_care\_ch5\_redu.phased --effective-size 11418 |
| --input-bed <.bed+.bim+.fam>  --input-map b37.map  --output-max output\_file(.haps & .sample) |
| /BiO/NIPT/Analysis/minjung/imputation/shapeit\_static/bin/shapeit -convert --input-haps Genome\_care\_ch1\_redu.phased --output-vcf Genome\_care\_ch1\_redu.phased.vcf |
| input file(.haps & .sample) convert to .vcf format |
| **bcftools** |
| /BiO/NIPT/Analysis/minjung/ngs/bcftools/bcftools norm --rm-dup both phased\_impu\_chr22.dose.vcf -o output.vcf  /BiO/NIPT/Analysis/minjung/ngs/bcftools/bcftools view --exclude-types indels phased\_impu\_chr22.dose.vcf -o output.vcf |
| --rm-dup : both -> indel, snp 에서 dup remove  --exclude-type indels : indels remove |

|  |
| --- |
| **sickle 명령어 : quality 와 bp 설정해서 원하는 값 선택할 수 있게 해줌.** |
| sickle se -f GC1611C3323\_20161126-470004\_IonXpress\_037.Fastq -t sanger -o GC1611C3323\_20161126-470004\_IonXpress\_037.trimmed.Fastq -q 20 -l 35 |
| sickle se : single end  -f : input file  -t : qual type  -o : output file  -q, -l : quality value, length value |
| **R command argument 사용법** |
| R CMD BATCH "--args --i=/BiO/NIPT/Analysis/minjung/ngs/seqFF/indir/ --f=GC1611C3352\_20161128-430707\_IonXpress\_020.sort.sam --d=/BiO/NIPT/Analysis/minjung/ngs/seqFF/outdir/ --o=GC1611C3352\_20161128-430707\_IonXpress\_020.ff --t=sam" SeqFF.R |
| Determination of fetal DNA fraction ~ seqFF.R 사용 command. |
| **R model : 선형회귀 모델** |
| data <- read.table(‘file’, header=FALSE)  attach(data)  model <- lm(V1~V2)  summary(model)  model |