Whole Genome Sequencing

El objetivo es definir un procedimiento que procese un numero grande de secuencias WGS en el menor tiempo posible. Para ello,una vez definido el pipeline deberemos automatizar las tareas e integrarlas en el schedule manager del cluster (Como usar el cluster sin morir en el intento).

Nota: Aunque el script de ejecución final está integrado en el cluster de Fundació ACE, puede ser modificado fácilmente para integrarlo en cualquier otro cluster que use SLURM y probablemente pueda modificarse para otros *schedule managers*.

tl;dr

```
./wgs.py -o <output_dir> [-cut <list.txt>] [-g] -s <input_dir>
```

Opciones:

- -s <input_dir> : (Mandatory) Directorio donde se encuentran todas las secuencias. El script buscara los sujetos y sus archivos dentro de este directorio.
- -o <output_dir> : (Opcional) Directorio donde se escribiran los resultados. En caso de obviarse se escribiran en el directorio desde el cual se lanza el script.
- -cut <list.txt> : (Opcional) Dice al script que analice SOLO los sujetos incluidos en el archivo que se suministra <list.txt>. Este archivo debe ser una listasimple de los sujetos a analizar.
- -g : Indica que no se borren los archivos temporales. Por defecto se borran, a no ser que se ponga este switch.

Ejemplo

```
[osotolongo@brick03 wgs]$ bin/wgs.py -o /home/osotolongo/wgs/temp -c
only.txt -s /the_dysk/BGI_exome/F18FTSEUET0180/HUMehbE/
Submitted batch job 1686
Submitted batch job 1687
Submitted batch job 1688
Submitted batch job 1689
Submitted batch job 1693
Submitted batch job 1694
Submitted batch job 1695
Submitted batch job 1696
Submitted batch job 1700
Submitted batch job 1701
Submitted batch job 1702
Submitted batch job 1703
Submitted batch job 1707
[osotolongo@brick03 wgs]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES
NODELIST(REASON)
              1691      fast verify_s osotolon PD       0:00      1
```

```
(Dependency)
              1692      fast validate osotolon PD       0:00      1
(Dependency)
              1698      fast verify_s osotolon PD       0:00      1
(Dependency)
              1699      fast validate osotolon PD       0:00      1
(Dependency)
              1705      fast verify_s osotolon PD       0:00      1
(Dependency)
              1706      fast validate osotolon PD       0:00      1
(Dependency)
              1707      fast  wgs_end osotolon PD       0:00      1
(Dependency)
              1690      fast sam_seq- osotolon  R       0:04      1 brick01
              1697      fast sam_seq- osotolon  R       0:01      1 brick01
              1704      fast sam_seq- osotolon  R       0:01      1 brick01
```

Pipeline

Primeramente hemos de definir el pipeline que se correra dentro del cluster. Aqui se ha de tener cuidado porque todas las herramientas y archivos han de ser accesibles desde cualquier nodo. En aras del siguiente paso podemos dividir el proceso en varios trozos. Tomemos por ejemplo el sujeto *seq-5*. Aqui los pasos son,

```
/nas/usr/local/bin/bwa mem -t 4 -R
"@RG\tID:V300016291_L01_549\tSM:seq5\tPL:BGI\tPI:380" -M
/the_dysk/BGI_exome/reference/Homo_sapiens_assembly38
/the_dysk/BGI_exome/F18FTSEUET0180/HUMehbE/seq-5/V300016291_L01_549_1.fq.gz
/the_dysk/BGI_exome/F18FTSEUET0180/HUMehbE/seq-5/V300016291_L01_549_2.fq.gz
> /the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5_0.sam

/nas/usr/local/bin/bwa mem -t 4 -R
"@RG\tID:V300016291_L01_550\tSM:seq5\tPL:BGI\tPI:380" -M
/the_dysk/BGI_exome/reference/Homo_sapiens_assembly38
/the_dysk/BGI_exome/F18FTSEUET0180/HUMehbE/seq-5/V300016291_L01_550_1.fq.gz
/the_dysk/BGI_exome/F18FTSEUET0180/HUMehbE/seq-5/V300016291_L01_550_2.fq.gz
> /the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5_1.sam

/nas/usr/local/bin/bwa mem -t 4 -R
"@RG\tID:V300016291_L01_551\tSM:seq5\tPL:BGI\tPI:380" -M
/the_dysk/BGI_exome/reference/Homo_sapiens_assembly38
/the_dysk/BGI_exome/F18FTSEUET0180/HUMehbE/seq-5/V300016291_L01_551_1.fq.gz
/the_dysk/BGI_exome/F18FTSEUET0180/HUMehbE/seq-5/V300016291_L01_551_2.fq.gz
> /the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5_2.sam

/nas/usr/local/bin/bwa mem -t 4 -R
"@RG\tID:V300016291_L01_552\tSM:seq5\tPL:BGI\tPI:380" -M
/the_dysk/BGI_exome/reference/Homo_sapiens_assembly38
/the_dysk/BGI_exome/F18FTSEUET0180/HUMehbE/seq-5/V300016291_L01_552_1.fq.gz
```

```
/the_dysk/BGI_exome/F18FTSEUET0180/HUMehbE/seq-5/V300016291_L01_552_2.fq.gz
> /the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5_3.sam

java -Djava.io.tmpdir=/nas/osotolongo/tmp/ -Xmx8g -jar
/nas/usr/local/bin/picard.jar MergeSamFiles
I=/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5_0.sam
I=/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5_1.sam
I=/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5_2.sam
I=/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5_3.sam
O=/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5.sam
java -Djava.io.tmpdir=/nas/osotolongo/tmp/ -Xmx8g -jar
/nas/usr/local/bin/picard.jar SortSam
I=/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5.sam
O=/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5_sorted.bam
SORT_ORDER=coordinate
/nas/software/samtools/bin/samtools index
/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5_sorted.bam

/nas/usr/local/bin/verifyBamID --vcf
/the_dysk/BGI_exome/reference/hapmap_3.3.hg38.vcf.gz --bam
/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5_sorted.bam --chip-none
--maxDepth 1000 --precise --verbose --ignoreRG --out
/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/results/seq5_verifybam |& grep
-v "Skipping marker"

java -Djava.io.tmpdir=/nas/osotolongo/tmp/ -Xmx8g -jar
/nas/usr/local/bin/picard.jar ValidateSamFile IGNORE=MATE_NOT_FOUND
IGNORE=MISSING_READ_GROUP IGNORE=RECORD_MISSING_READ_GROUP
I=/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5_sorted.bam
java -Djava.io.tmpdir=/nas/osotolongo/tmp/ -Xmx8g -jar
/nas/usr/local/bin/picard.jar MarkDuplicates
I=/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5_sorted.bam
O=/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5_GATKready.bam
METRICS_FILE=/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5_metrics.t
xt QUIET=true MAX_RECORDS_IN_RAM=2000000 ASSUME_SORTED=TRUE
CREATE_INDEX=TRUE
java -jar /nas/usr/local/opt/gatk3/GenomeAnalysisTK.jar -T DepthOfCoverage -
R /the_dysk/BGI_exome/reference/Homo_sapiens_assembly38.fasta -nt 1 -ct 10 -
ct 15 -ct 20 -ct 30 --omitDepthOutputAtEachBase --omitIntervalStatistics --
omitLocusTable -L
/the_dysk/BGI_exome/reference/MGI_Exome_Capture_V5_bis2.bed -I
/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5_GATKready.bam -o
/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/results/seq5_exome_coverage
singularity run --cleanenv -B /nas:/nas -B /the_dysk:/the_dysk
/usr/local/bin/gatk4.simg gatk --java-options "-
DGATK_STACKTRACE_ON_USER_EXCEPTION=true -Xmx16G" BaseRecalibrator -I
/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5_GATKready.bam -R
/the_dysk/BGI_exome/reference/Homo_sapiens_assembly38.fasta --known-sites
/the_dysk/BGI_exome/reference/Mills_and_1000G_gold_standard.indels.hg38.vcf.
gz --known-sites /the_dysk/BGI_exome/reference/dbsnp_146.hg38.vcf.gz --
known-sites
```

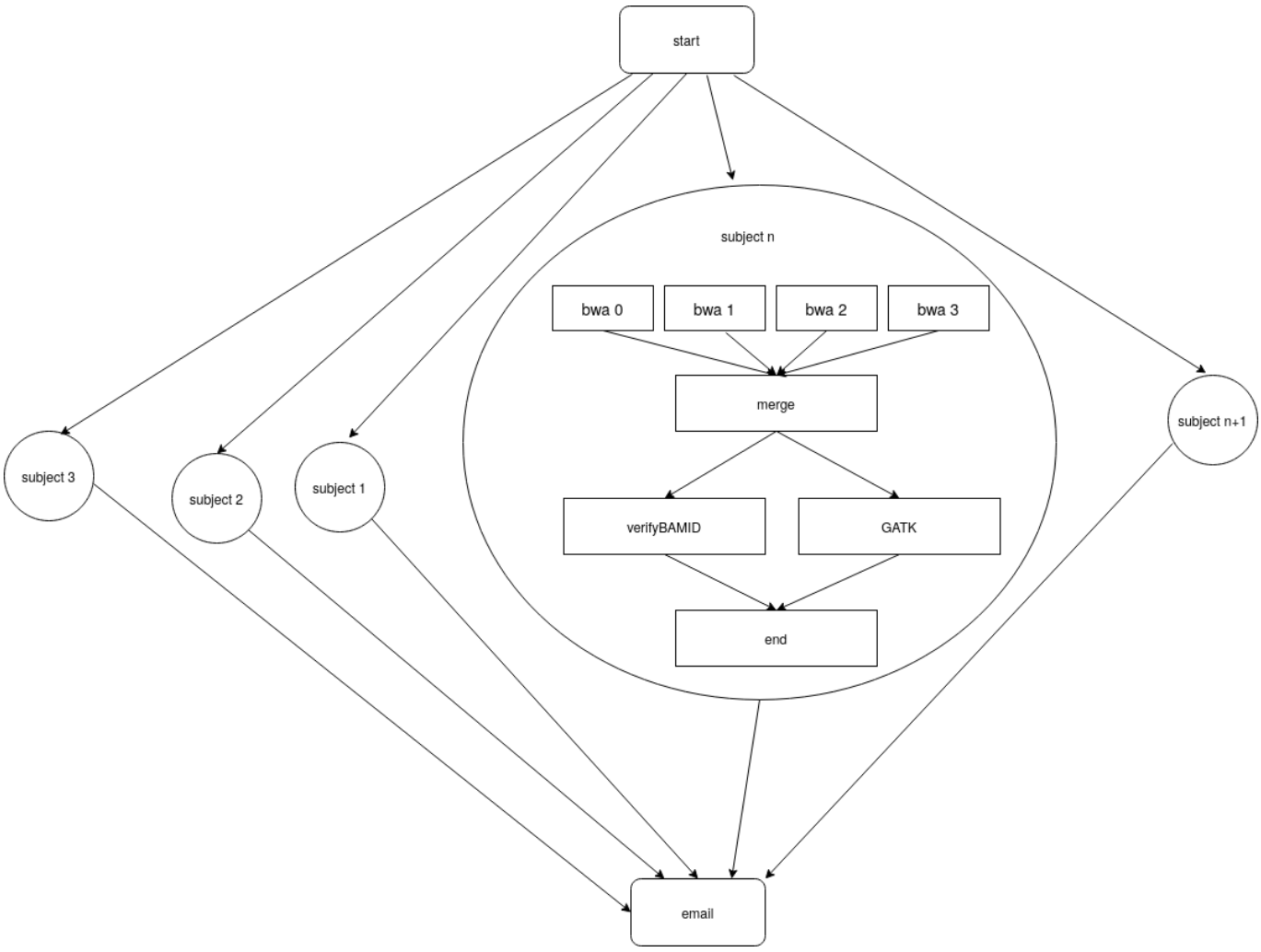
```
/the_dysk/BGI_exome/reference/1000G_phase1.snps.high_confidence.hg38.vcf.gz
-O /the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5_recal_data.table1
singularity run --cleanenv -B /nas:/nas -B /the_dysk:/the_dysk
/usr/local/bin/gatk4.simg gatk --java-options "-
DGATK_STACKTRACE_ON_USER_EXCEPTION=true -Xmx16G" ApplyBQSR -R
/the_dysk/BGI_exome/reference/Homo_sapiens_assembly38.fasta -I
/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5_GATKready.bam -bqsr-
recal-file
/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5_recal_data.table1 -O
/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/results/seq5_recal.bam
singularity run --cleanenv -B /nas:/nas -B /the_dysk:/the_dysk
/usr/local/bin/gatk4.simg gatk --java-options "-
DGATK_STACKTRACE_ON_USER_EXCEPTION=true -Xmx16G" AnalyzeCovariates -bqsr
/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5_recal_data.table1 --
plots
/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/results/seq5_AnalyzeCovariates.
pdf
singularity run --cleanenv -B /nas:/nas -B /the_dysk:/the_dysk
/usr/local/bin/gatk4.simg gatk --java-options "-
DGATK_STACKTRACE_ON_USER_EXCEPTION=true -Xmx16G" BaseRecalibrator -I
/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/results/seq5_recal.bam -R
/the_dysk/BGI_exome/reference/Homo_sapiens_assembly38.fasta --known-sites
/the_dysk/BGI_exome/reference/Mills_and_1000G_gold_standard.indels.hg38.vcf.
gz --known-sites /the_dysk/BGI_exome/reference/dbsnp_146.hg38.vcf.gz --
known-sites
/the_dysk/BGI_exome/reference/1000G_phase1.snps.high_confidence.hg38.vcf.gz
-O /the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5_recal_data.table2
singularity run --cleanenv -B /nas:/nas -B /the_dysk:/the_dysk
/usr/local/bin/gatk4.simg gatk --java-options "-
DGATK_STACKTRACE_ON_USER_EXCEPTION=true -Xmx16G" AnalyzeCovariates -before
/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5_recal_data.table1 -
after
/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/seq5_recal_data.table2 -
plots /the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/results/seq5_before-
after-plots.pdf
singularity run --cleanenv -B /nas:/nas -B /the_dysk:/the_dysk
/usr/local/bin/gatk4.simg gatk --java-options "-
DGATK_STACKTRACE_ON_USER_EXCEPTION=true -Xmx16G" HaplotypeCaller -R
/the_dysk/BGI_exome/reference/Homo_sapiens_assembly38.fasta -I
/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/results/seq5_recal.bam -ERC
GVCF --dbsnp /the_dysk/BGI_exome/reference/dbsnp_146.hg38.vcf.gz -O
/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/results/seq5_raw.snps.indels.g.
vcf.gz
singularity run --cleanenv -B /nas:/nas -B /the_dysk:/the_dysk
/usr/local/bin/gatk4.simg gatk --java-options "-
DGATK_STACKTRACE_ON_USER_EXCEPTION=true -Xmx16G -XX:+UseConcMarkSweepGC"
VariantEval -R /the_dysk/BGI_exome/reference/Homo_sapiens_assembly38.fasta -
L /the_dysk/BGI_exome/reference/MGI_Exome_Capture_V5_bis2.bed -D
/the_dysk/BGI_exome/reference/dbsnp_146.hg38.vcf.gz -O
/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/results/seq5_eval.gatkreport --
eval
```

```
/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/results/seq5_raw.snps.indels.g.vcf.gz
```

Este pipeline se ha dividido en varios bloques que dependen unos de otros. Los 4 primeros son independientes y se pueden correr en paralelo. El numero 5 depende de la terminacion de los 4 primeros. Los bloques 6 y 7 dependen del numero 5 pero pueden correrse independientemente.

Paralelizacion

El montaje correcto del pipeline permite la paralelizacioen del procediemiento dentro de cada secuenciacion, pero ademas, debe paralelizarse el procedimientp total. Es decir, cadda sujeto debe correr en paralelo a los demas. Para ello basta crear un sistema de dependencias como el que se muestra en la figura.



Entonces, una vez definido el pipeline, la secuencia de ejecucion de cada trozo y el modo de paralelizacion seria sencillo definir el flujo de ejecucion si supieramos que archivos iniciales debe cargar cada ejecucion inicial (los *bwa*).

Parsing

Lo primero que ddebemos saber es que el directorio de *input* esta compuesto por una lista de subdirectorios cada uno perteneciente a un sujeto distinto.

```
[osotolongo@detritus HUMehbE]$ pwd
/the_dysk/BGI_exome/F18FTSEUET0180/HUMehbE
[osotolongo@detritus HUMehbE]$ ls
BGI_Data_List_F18FTSEUET0180_filled.pdf                    md5sum.check
seq-1   seq-11  seq-13  seq-15  seq-17  seq-19  seq-20  seq-22  seq-24
seq-26  seq-28  seq-4  seq-6  seq-8
BGI_Sequencing_Analysis_Report_F18FTSEUET0180_HUMehbE.pdf  md5sum.txt
seq-10  seq-12  seq-14  seq-16  seq-18  seq-2   seq-21  seq-23  seq-25
seq-27  seq-3   seq-5  seq-7  seq-9
```

Para cada sujeto hay una lista de ocho archivos que deben parearse segun el nombre de archivo.

```
[osotolongo@detritus HUMehbE]$ ls seq-5/
V300016291_L01_549_1.fq.gz  V300016291_L01_550_1.fq.gz
V300016291_L01_551_1.fq.gz  V300016291_L01_552_1.fq.gz
V300016291_L01_549_2.fq.gz  V300016291_L01_550_2.fq.gz
V300016291_L01_551_2.fq.gz  V300016291_L01_552_2.fq.gz
```

En este ejemplo deben parearse,

- V300016291_L01_549_1.fq.gz y V300016291_L01_549_2.fq.gz
- V300016291_L01_550_1.fq.gz y V300016291_L01_550_2.fq.gz
- V300016291_L01_551_1.fq.gz y V300016291_L01_551_2.fq.gz
- V300016291_L01_552_1.fq.gz y V300016291_L01_552_2.fq.gz

asi que la estructura a seguir es bastante clara. No obstante a que los nombres de archivo varian entre sujetos, siguen la misma estructura.

```
[osotolongo@detritus HUMehbE]$ ls seq-10
V300016281_L01_553_1.fq.gz  V300016281_L01_554_1.fq.gz
V300016281_L01_555_1.fq.gz  V300016281_L01_556_1.fq.gz
V300016281_L01_553_2.fq.gz  V300016281_L01_554_2.fq.gz
V300016281_L01_555_2.fq.gz  V300016281_L01_556_2.fq.gz
```

Lo que voy hacer entonces es definir un *hash* donde se guarda la informacion relativa a cada sujeto (incluidos los patrones en los archivos originales) y a partir de ahi construir los scripts.

Veamos, primero leo el *input dir*

```
lpath = os.path.abspath(src_dir)
dir_cont = next(os.walk(src_dir))[1]
```

luego, en dependencia de las opciones puedo reducir la lista o no,

```
if cfile and os.path.isfile(cfile):
```

```
  f = open(cfile, 'r')
  cuts = f.read().splitlines()
  dir_cont = set(dir_cont) - set(cuts)
```

y ahora lleno el *hash* de datos,

```
for pollo in dir_cont:
  fq_list = []
  fq_files = next(os.walk(lpath+'/'+pollo))[2]
  for fq in fq_files:
    ids = re.search(r"^V(\d*?)\_L(\d*?)\_(\d*?)\_\d\.fq\.gz$", fq)
    fq_name ='V'+ids.group(1)+'_L'+ids.group(2)
    fq_list.append(ids.group(3))
```

Programatic tree

Ya casi esta, ahora por cada sujeto guardado pueden definirse los primeros 4 procesos,

```
  for i in range(0,4):
    fqid = fq_list[2*i]
    orderfile = outdir+'/bwa_'+pollo+'_'+str(i)+'.sh'
    ord_content = '#!/bin/bash\n'
    ord_content += '#SBATCH -J sam_'+pollo+'\n'
    ord_content += '#SBATCH --time=24:0:0\n'
    ord_content += '#SBATCH -o '+outdir+'/bwa_'+pollo+'-%j'+'\n'
    ord_content += '#SBATCH -c 8\n'
    ord_content += '#SBATCH --mem-per-cpu=4G\n'
    ord_content += '#SBATCH -p fast\n'
    ord_content += '#SBATCH --mail-type=FAIL,TIME_LIMIT,STAGE_OUT\n'
    ord_content += '#SBATCH --mail-user='+os.environ.get('USER')+'\n'
    ord_content += bwa+' mem  -t 4 -R
"@RG\tID:'+fq_name+'_'+fqid+'\tSM:'+pname+'\tPL:BGI\tPI:380" -M
'+ref_dir+'/Homo_sapiens_assembly38
'+src_dir+'/'+pollo+'/'+fq_name+'_'+fqid+'_1.fq.gz
'+src_dir+'/'+pollo+'/'+fq_name+'_'+fqid+'_2.fq.gz >
'+tmpdir+'/'+pname+_+str(i)+'.sam\n'
    osf = open(orderfile, 'w')
    osf.write(ord_content)
    osf.close()
    gsconv+= 'I='+tmpdir+'/'+pname+'_'+str(i)+'.sam '
    os.system('sbatch '+orderfile)
```

El numero 5 se hace que dependa de estos 4,

```
  orderfile = outdir+'/merge_'+pollo+'.sh'
  ord_content = '#!/bin/bash\n'
  ord_content += '#SBATCH -J sam_'+pollo+'\n'
```

```
  ord_content += '#SBATCH --time=24:0:0\n'
  ord_content += '#SBATCH -o '+outdir+'/merge_'+pollo+'-%j\n'
  ord_content += '#SBATCH -c 8\n'
  ord_content += '#SBATCH --mem-per-cpu=4G\n'
  ord_content += '#SBATCH -p fast\n'
  ord_content += '#SBATCH --mail-type=FAIL,TIME_LIMIT,STAGE_OUT\n'
  ord_content += '#SBATCH --mail-user='+os.environ.get('USER')+'\n'
  ord_content += picard+' MergeSamFiles '+gsconv+'
O='+tmpdir+'/'+pname+'.sam\n'
  ord_content += picard+' SortSam I='+tmpdir+'/'+pname+'.sam
O='+tmpdir+'/'+pname+'_sorted.bam SORT_ORDER=coordinate\n'
  ord_content += samtools+' index '+tmpdir+'/'+pname+'_sorted.bam\n'
  osf = open(orderfile, 'w')
  osf.write(ord_content)
  osf.close()
  ujobid = subprocess.getoutput('sbatch --dependency=singleton --parsable
'+orderfile)
```

El 6 y el 7 dependen del 5,

```
  orderfile = outdir+'/verify_'+pollo+'.sh'
  ord_content = '#!/bin/bash\n'
  ord_content += '#SBATCH -J verify_'+pollo+'\n'
  ord_content += '#SBATCH --time=24:0:0\n'
  ord_content += '#SBATCH -o '+outdir+'/verify_'+pollo+'-%j\n'
  ord_content += '#SBATCH -c 8\n'
  ord_content += '#SBATCH --mem-per-cpu=4G\n'
  ord_content += '#SBATCH -p fast\n'
  ord_content += '#SBATCH --mail-type=FAIL,TIME_LIMIT,STAGE_OUT\n'
  ord_content += '#SBATCH --mail-user='+os.environ.get('USER')+'\n'
  ord_content += verifybamib+' --vcf '+ref_dir+'/hapmap_3.3.hg38.vcf.gz --
bam '+tmpdir+'/'+pname+'_sorted.bam --chip-none --maxDepth 1000 --precise --
verbose --ignoreRG --out '+resdir+'/'+pname+'_verifybam |& grep -v "Skipping
marker"\n'
  osf = open(orderfile, 'w')
  osf.write(ord_content)
  osf.close()
  jobid = subprocess.getoutput('sbatch --parsable --
dependency=afterok:'+ujobid+' '+orderfile)
  jobids.append(jobid)
```

```
  orderfile = outdir+'/validate_'+pollo+'.sh'
  ord_content = '#!/bin/bash\n'
  ord_content += '#SBATCH -J validate_'+pollo+'\n'
  ord_content += '#SBATCH --time=24:0:0\n'
  ord_content += '#SBATCH -o '+outdir+'/validate_'+pollo+'-%j\n'
  ord_content += '#SBATCH -c 8\n'
  ord_content += '#SBATCH --mem-per-cpu=4G\n'
  ord_content += '#SBATCH -p fast\n'
```

```
  ord_content += '#SBATCH --mail-type=FAIL,TIME_LIMIT,STAGE_OUT\n'
  ord_content += '#SBATCH --mail-user='+os.environ.get('USER')+'\n'
  ord_content += picard+' ValidateSamFile IGNORE=MATE_NOT_FOUND
IGNORE=MISSING_READ_GROUP IGNORE=RECORD_MISSING_READ_GROUP
I='+tmpdir+'/'+pname+'_sorted.bam\n'
  ord_content += picard+' MarkDuplicates I='+tmpdir+'/'+pname+'_sorted.bam
O='+tmpdir+'/'+pname+'_GATKready.bam
METRICS_FILE='+resdir+'/'+pname+'_metrics.txt QUIET=true
MAX_RECORDS_IN_RAM=2000000 ASSUME_SORTED=TRUE CREATE_INDEX=TRUE\n'
  ord_content += gatk3+' -T DepthOfCoverage -R
'+ref_dir+'/Homo_sapiens_assembly38.fasta -nt 1 -ct 10 -ct 15 -ct 20 -ct 30
--omitDepthOutputAtEachBase --omitIntervalStatistics --omitLocusTable -L
'+ref_dir+'/MGI_Exome_Capture_V5_bis2.bed -I
'+tmpdir+'/'+pname+'_GATKready.bam -o '+resdir+'/'+pname+'_exome_coverage\n'
  ord_content += gatk4+' BaseRecalibrator -I
'+tmpdir+'/'+pname+'_GATKready.bam -R
'+ref_dir+'/Homo_sapiens_assembly38.fasta --known-sites
'+ref_dir+'/Mills_and_1000G_gold_standard.indels.hg38.vcf.gz --known-sites
'+ref_dir+'/dbsnp_146.hg38.vcf.gz --known-sites
'+ref_dir+'/1000G_phase1.snps.high_confidence.hg38.vcf.gz -O
'+resdir+'/'+pname'_recal_data.table1\n'
  ord_content += gatk4+' ApplyBQSR -R
'+ref_dir+'/Homo_sapiens_assembly38.fasta -I
'+tmpdir+'/'+pname+'_GATKready.bam -bqsr-recal-file
'+resdir+'/'+pname+'_recal_data.table1 -O '+resdir+'/'+pname+'_recal.bam\n'
  ord_content += gatk4+' AnalyzeCovariates -bqsr
'+resdir+'/'+pname+'_recal_data.table1 --plots
'+resdir+'/'+pname+'_AnalyzeCovariates.pdf\n'
  ord_content += gatk4+' BaseRecalibrator -I '+resdir+'/'+pname+'_recal.bam
-R '+ref_dir+'/Homo_sapiens_assembly38.fasta --known-sites
'+ref_dir+'/Mills_and_1000G_gold_standard.indels.hg38.vcf.gz --known-sites
'+ref_dir+'/dbsnp_146.hg38.vcf.gz --known-sites
'+ref_dir+'/1000G_phase1.snps.high_confidence.hg38.vcf.gz -O
'+resdir+'/'+pname+'_recal_data.table2\n'
  ord_content += gatk4+' AnalyzeCovariates -before
'+resdir+'/'+pname+'_recal_data.table1 -after
'+resdir+'/'+pname+'_recal_data.table2 -plots '+resdir+'/'+pname+'_before-
after-plots.pdf\n'
  ord_content += gatk4+' HaplotypeCaller -R
'+ref_dir+'/Homo_sapiens_assembly38.fasta -I '+resdir+'/'+pname+'_recal.bam
-ERC GVCF --dbsnp '+ref_dir+'/dbsnp_146.hg38.vcf.gz -O
'+resdir+'/'+pname+'_raw.snps.indels.g.vcf.gz\n'
  ord_content += gatk4_l+' VariantEval -R
'+ref_dir+'/Homo_sapiens_assembly38.fasta -L
'+ref_dir+'/MGI_Exome_Capture_V5_bis2.bed -D
'+ref_dir+'/dbsnp_146.hg38.vcf.gz -O '+resdir+'/'+pname+'_eval.gatkreport --
eval '+resdir+'/'+pname+'_raw.snps.indels.g.vcf.gz\n'
  osf = open(orderfile, 'w')
  osf.write(ord_content)
  osf.close()
  jobid = subprocess.getoutput('sbatch --parsable --
```

```
dependency=afterok:'+ujobid+' '+orderfile)
  jobids.append(jobid)
```

y por ultimo, hay un proceso que depende de que terminen todos los demas (TODOS) y lo que hace es limpiar los archivos temporales y enviar un email de finalizacion,

```
orderfile = outdir+'/wgs_end.sh'
ord_content = '#!/bin/bash\n'
ord_content += '#SBATCH -J wgs_end\n'
ord_content += '#SBATCH -o '+outdir+'/wgs_end-%j\n'
ord_content += '#SBATCH --mail-type=END\n'
ord_content += '#SBATCH --mail-user='+os.environ.get('USER')+'\n'
if(debug):
  ord_content += 'rm -rf '+wdir+'/*/tmp\n'
else:
  ord_content += ':\n'
osf = open(orderfile, 'w')
osf.write(ord_content)
osf.close()
sjobs = ','.join(map(str,jobids))
os.system('sbatch --depend=afterok:'+sjobs+' '+orderfile)
```

La magia completa aqui, en menos de 200 lineas

wgs.py

```
#!/usr/bin/python3

"""
Copyright 2020 O. Sotolongo <asqwerty@gmail.com>

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.
"""

import sys
import os
import getopt
import re

"""
```

```
Rellenar aqui los paths del sistema
"""
ref_dir = '/the_dysk/BGI_exome/reference'
bwa = '/nas/usr/local/bin/bwa'
picard = 'java -Djava.io.tmpdir=/nas/'+os.environ.get('USER')+'/tmp/ -
Xmx8g -jar /nas/usr/local/bin/picard.jar'
samtools = '/nas/software/samtools/bin/samtools'
verifybamib = '/nas/usr/local/bin/verifyBamID'
gatk3 = 'java -jar /nas/usr/local/opt/gatk3/GenomeAnalysisTK.jar'
gatk4 = 'singularity run --cleanenv -B /nas:/nas -B /the_dysk:/the_dysk
/usr/local/bin/gatk4.simg gatk --java-options "-
DGATK_STACKTRACE_ON_USER_EXCEPTION=true -Xmx16G"'
gatk4_l = 'singularity run --cleanenv -B /nas:/nas -B
/the_dysk:/the_dysk /usr/local/bin/gatk4.simg gatk --java-options "-
DGATK_STACKTRACE_ON_USER_EXCEPTION=true -Xmx16G -
XX:+UseConcMarkSweepGC"'
"""
No tocar nada debajo de esto si no sabes lo que estas haciendo
"""

# Get CLI inputs
short_args = 'c:o:gs:'
long_args = ['cut=', 'output=', 'debug', 'source=']
debug = 0
cfile=''
outdatadir=''

try:
  args, values = getopt.getopt(sys.argv[1:], short_args, long_args)
except getopt.error as err:
  print(str(err))
  sys.exit(2)

for a,v in args:
  if a in ('--cut', '-c'):
    cfile = v
  elif a in ('--output', '-o'):
    outdatadir = v
  elif a in ('--debug', '-g'):
    debug = 1
  elif a in ('--source', '-s'):
    src_dir = v

lpath = os.path.abspath(src_dir)
dir_cont = next(os.walk(src_dir))[1]

if cfile and os.path.isfile(cfile):
  f = open(cfile, 'r')
  cuts = f.read().splitlines()
  dir_cont = set(dir_cont) - set(cuts)
```

```
# Creo el entorno de ejecucion
tmp_shit = '/nas/'+os.environ.get('USER')+'/tmp/'
if not os.path.isdir(tmp_shit): os.mkdir(tmp_shit)
if outdatadir:
  os.mkdir(outdatadir)
  wdir = outdatadir
else:
  wdir = os.environ.get('PWD')
outdir = wdir+'/slurm'
if not os.path.isdir(outdir): os.mkdir(outdir)
fq = {}
for pollo in dir_cont:
  fq_list = []
  fq_files = next(os.walk(lpath+'/'+pollo))[2]
  for fq in fq_files:
    ids = re.search(r"^V(\d*?)\_L(\d*?)\_(\d*?)\_\d\.fq\.gz$", fq)
    fq_name ='V'+ids.group(1)+'_L'+ids.group(2)
    fq_list.append(ids.group(3))
  udir = wdir+'/'+pollo
  if not os.path.isdir(udir): os.mkdir(udir)
  tmpdir = udir+'/tmp'
  if not os.path.isdir(tmpdir): os.mkdir(tmpdir)
  resdir = udir+'/results'
  if not os.path.isdir(resdir): os.mkdir(resdir)
  jobids = []
  gsconv = ''
  pname = re.sub('-','',pollo)
  for i in range(0,4):
    fqid = fq_list[2*i]
    orderfile = outdir+'/bwa_'+pollo+'_'+str(i)+'.sh'
    ord_content = '#!/bin/bash\n'
    ord_content += '#SBATCH -J sam_'+pollo+'\n'
    ord_content += '#SBATCH --time=24:0:0\n'
    ord_content += '#SBATCH -o '+outdir+'/bwa_'+pollo+'-%j'+'\n'
    ord_content += '#SBATCH -c 8\n'
    ord_content += '#SBATCH --mem-per-cpu=4G\n'
    ord_content += '#SBATCH -p fast\n'
    ord_content += '#SBATCH --mail-type=FAIL,TIME_LIMIT,STAGE_OUT\n'
    ord_content += '#SBATCH --mail-user='+os.environ.get('USER')+'\n'
    ord_content += bwa+' mem  -t 4 -R
"@RG\tID:'+fq_name+'_'+fqid+'\tSM:'+pname+'\tPL:BGI\tPI:380" -M
'+ref_dir+'/Homo_sapiens_assembly38
'+src_dir+'/'+pollo+'/'+fq_name+'_'+fqid+'_1.fq.gz
'+src_dir+'/'+pollo+'/'+fq_name+'_'+fqid+'_2.fq.gz >
'+tmpdir+'/'+pname+_+str(i)+'.sam\n'
    osf = open(orderfile, 'w')
    osf.write(ord_content)
    osf.close()
    gsconv+= 'I='+tmpdir+'/'+pname+'_'+str(i)+'.sam '
    os.system('sbatch '+orderfile)
  orderfile = outdir+'/merge_'+pollo+'.sh'
```

```
  ord_content = '#!/bin/bash\n'
  ord_content += '#SBATCH -J sam_'+pollo+'\n'
  ord_content += '#SBATCH --time=24:0:0\n'
  ord_content += '#SBATCH -o '+outdir+'/merge_'+pollo+'-%j\n'
  ord_content += '#SBATCH -c 8\n'
  ord_content += '#SBATCH --mem-per-cpu=4G\n'
  ord_content += '#SBATCH -p fast\n'
  ord_content += '#SBATCH --mail-type=FAIL,TIME_LIMIT,STAGE_OUT\n'
  ord_content += '#SBATCH --mail-user='+os.environ.get('USER')+'\n'
  ord_content += picard+' MergeSamFiles '+gsconv+'
O='+tmpdir+'/'+pname+'.sam\n'
  ord_content += picard+' SortSam I='+tmpdir+'/'+pname+'.sam
O='+tmpdir+'/'+pname+'_sorted.bam SORT_ORDER=coordinate\n'
  ord_content += samtools+' index '+tmpdir+'/'+pname+'_sorted.bam\n'
  osf = open(orderfile, 'w')
  osf.write(ord_content)
  osf.close()
  ujobid = subprocess.getoutput('sbatch --dependency=singleton --
parsable '+orderfile)
  orderfile = outdir+'/verify_'+pollo+'.sh'
  ord_content = '#!/bin/bash\n'
  ord_content += '#SBATCH -J verify_'+pollo+'\n'
  ord_content += '#SBATCH --time=24:0:0\n'
  ord_content += '#SBATCH -o '+outdir+'/verify_'+pollo+'-%j\n'
  ord_content += '#SBATCH -c 8\n'
  ord_content += '#SBATCH --mem-per-cpu=4G\n'
  ord_content += '#SBATCH -p fast\n'
  ord_content += '#SBATCH --mail-type=FAIL,TIME_LIMIT,STAGE_OUT\n'
  ord_content += '#SBATCH --mail-user='+os.environ.get('USER')+'\n'
  ord_content += verifybamib+' --vcf '+ref_dir+'/hapmap_3.3.hg38.vcf.gz
--bam '+tmpdir+'/'+pname+'_sorted.bam --chip-none --maxDepth 1000 --
precise --verbose --ignoreRG --out '+resdir+'/'+pname+'_verifybam |&
grep -v "Skipping marker"\n'
  osf = open(orderfile, 'w')
  osf.write(ord_content)
  osf.close()
  jobid = subprocess.getoutput('sbatch --parsable --
dependency=afterok:'+ujobid+' '+orderfile)
  jobids.append(jobid)
  orderfile = outdir+'/validate_'+pollo+'.sh'
  ord_content = '#!/bin/bash\n'
  ord_content += '#SBATCH -J validate_'+pollo+'\n'
  ord_content += '#SBATCH --time=24:0:0\n'
  ord_content += '#SBATCH -o '+outdir+'/validate_'+pollo+'-%j\n'
  ord_content += '#SBATCH -c 8\n'
  ord_content += '#SBATCH --mem-per-cpu=4G\n'
  ord_content += '#SBATCH -p fast\n'
  ord_content += '#SBATCH --mail-type=FAIL,TIME_LIMIT,STAGE_OUT\n'
  ord_content += '#SBATCH --mail-user='+os.environ.get('USER')+'\n'
  ord_content += picard+' ValidateSamFile IGNORE=MATE_NOT_FOUND
IGNORE=MISSING_READ_GROUP IGNORE=RECORD_MISSING_READ_GROUP
```

```
I='+tmpdir+'/'+pname+'_sorted.bam\n'
  ord_content += picard+' MarkDuplicates
I='+tmpdir+'/'+pname+'_sorted.bam O='+tmpdir+'/'+pname+'_GATKready.bam
METRICS_FILE='+resdir+'/'+pname+'_metrics.txt QUIET=true
MAX_RECORDS_IN_RAM=2000000 ASSUME_SORTED=TRUE CREATE_INDEX=TRUE\n'
  ord_content += gatk3+' -T DepthOfCoverage -R
'+ref_dir+'/Homo_sapiens_assembly38.fasta -nt 1 -ct 10 -ct 15 -ct 20 -
ct 30 --omitDepthOutputAtEachBase --omitIntervalStatistics --
omitLocusTable -L '+ref_dir+'/MGI_Exome_Capture_V5_bis2.bed -I
'+tmpdir+'/'+pname+'_GATKready.bam -o
'+resdir+'/'+pname+'_exome_coverage\n'
  ord_content += gatk4+' BaseRecalibrator -I
'+tmpdir+'/'+pname+'_GATKready.bam -R
'+ref_dir+'/Homo_sapiens_assembly38.fasta --known-sites
'+ref_dir+'/Mills_and_1000G_gold_standard.indels.hg38.vcf.gz --known-
sites '+ref_dir+'/dbsnp_146.hg38.vcf.gz --known-sites
'+ref_dir+'/1000G_phase1.snps.high_confidence.hg38.vcf.gz -O
'+resdir+'/'+pname'_recal_data.table1\n'
  ord_content += gatk4+' ApplyBQSR -R
'+ref_dir+'/Homo_sapiens_assembly38.fasta -I
'+tmpdir+'/'+pname+'_GATKready.bam -bqsr-recal-file
'+resdir+'/'+pname+'_recal_data.table1 -O
'+resdir+'/'+pname+'_recal.bam\n'
  ord_content += gatk4+' AnalyzeCovariates -bqsr
'+resdir+'/'+pname+'_recal_data.table1 --plots
'+resdir+'/'+pname+'_AnalyzeCovariates.pdf\n'
  ord_content += gatk4+' BaseRecalibrator -I
'+resdir+'/'+pname+'_recal.bam -R
'+ref_dir+'/Homo_sapiens_assembly38.fasta --known-sites
'+ref_dir+'/Mills_and_1000G_gold_standard.indels.hg38.vcf.gz --known-
sites '+ref_dir+'/dbsnp_146.hg38.vcf.gz --known-sites
'+ref_dir+'/1000G_phase1.snps.high_confidence.hg38.vcf.gz -O
'+resdir+'/'+pname+'_recal_data.table2\n'
  ord_content += gatk4+' AnalyzeCovariates -before
'+resdir+'/'+pname+'_recal_data.table1 -after
'+resdir+'/'+pname+'_recal_data.table2 -plots
'+resdir+'/'+pname+'_before-after-plots.pdf\n'
  ord_content += gatk4+' HaplotypeCaller -R
'+ref_dir+'/Homo_sapiens_assembly38.fasta -I
'+resdir+'/'+pname+'_recal.bam -ERC GVCF --dbsnp
'+ref_dir+'/dbsnp_146.hg38.vcf.gz -O
'+resdir+'/'+pname+'_raw.snps.indels.g.vcf.gz\n'
  ord_content += gatk4_l+' VariantEval -R
'+ref_dir+'/Homo_sapiens_assembly38.fasta -L
'+ref_dir+'/MGI_Exome_Capture_V5_bis2.bed -D
'+ref_dir+'/dbsnp_146.hg38.vcf.gz -O
'+resdir+'/'+pname+'_eval.gatkreport --eval
'+resdir+'/'+pname+'_raw.snps.indels.g.vcf.gz\n'
  osf = open(orderfile, 'w')
  osf.write(ord_content)
  osf.close()
```

```
  jobid = subprocess.getoutput('sbatch --parsable --
dependency=afterok:'+ujobid+' '+orderfile)
  jobids.append(jobid)
orderfile = outdir+'/wgs_end.sh'
ord_content = '#!/bin/bash\n'
ord_content += '#SBATCH -J wgs_end\n'
ord_content += '#SBATCH -o '+outdir+'/wgs_end-%j\n'
ord_content += '#SBATCH --mail-type=END\n'
ord_content += '#SBATCH --mail-user='+os.environ.get('USER')+'\n'
if(debug):
  ord_content += 'rm -rf '+wdir+'/*/tmp\n'
else:
  ord_content += ':\n'
osf = open(orderfile, 'w')
osf.write(ord_content)
osf.close()
sjobs = ','.join(map(str,jobids))
os.system('sbatch --depend=afterok:'+sjobs+' '+orderfile)
```

Ejecucion

La forma correcta de ejecutar el script es,

```
$ ./wgs.py -o /the_dysk/BGI_exome/F18FTSEUET0180/WGS -s
/the_dysk/BGI_exome/F18FTSEUET0180/HUMehbE/
```

La opción *-o* indica al script donde guardar los resultados, en este caso en */the_dysk/BGI_exome/F18FTSEUET0180/WGS* . Los datos se leeran desde */the_dysk/BGI_exome/F18FTSEUET0180/HUMehbE/* ,

La opcion *-o* no es obligatoria. En caso de no suministrar un directorio de salida, esta se escribira en el directorio actual desde donde se ejeuta el script.

Los resultados se escriben en el directorio de salida, haciendo un subdirectorio para cada sujeto analizado y siguiendo las mismas convenciones del *input*,

```
[osotolongo@brick03 WGS]$ ls /the_dysk/BGI_exome/F18FTSEUET0180/WGS
seq-1   seq-11    seq-13    seq-15    seq-17    seq-19    seq-20    seq-22
seq-24    seq-26    seq-28    seq-4  seq-6  seq-8  slurm
```

```
seq-10  seq-12    seq-14    seq-16    seq-18    seq-2    seq-21    seq-23
seq-25    seq-27    seq-3    seq-5  seq-7  seq-9
```

Dentro de cada sujeto se crea u n directorio *results* con los resultados del analisis,

```
[osotolongo@brick03 WGS]$ tree seq-1
seq-1
└── results
    ├── seq1_AnalyzeCovariates.pdf
    ├── seq1_before-after-plots.pdf
    ├── seq1_eval.gatkreport
    ├── seq1_exome_coverage.sample_statistics
    ├── seq1_exome_coverage.sample_summary
    ├── seq1_metrics.txt
    ├── seq1_raw.snps.indels.g.vcf.gz
    ├── seq1_raw.snps.indels.g.vcf.gz.tbi
    ├── seq1_recal.bai
    ├── seq1_recal.bam
    ├── seq1_recal_data.table1
    ├── seq1_recal_data.table2
    ├── seq1_verifybam.depthSM
    ├── seq1_verifybam.log
    └── seq1_verifybam.selfSM

1 directory, 15 files
```

---

trick

Esta estructura permitiria ejecutar,

```
$ ./wgs.pl -o /the_dysk/BGI_exome/F18FTSEUET0180/HUMehbE -s
/the_dysk/BGI_exome/F18FTSEUET0180/HUMehbE
```

y **en caso de no tener problemas con los permisos**, los resultados quedarian en un directorio dentro del mismo sujeto de proyecto. No se ha hecho asi por defecto previendo precisamente los problemas de permisos.

Troubleshooting

Pueden ocurrir multitud de problemas en la ejecucion del script. Afortunadamente un fallo en uno de las tareas afecta solamente al sujeto individual sobre el que se realiza la tarea. El resto de sujetos se procesara correctamente.

Al ocurrir un fallo en algun script, el sistema enviara un email informando del fallo y el *jobid* asociado. Toda la informacion ira en el *subject* del email.

ejemplo 1

Aqui se informa que la tarea con *jobid = 154540* ha fallado. Tambien sabemos que esta relacionada con el sujeto *seq-27*.

```
SLURM Job_id=154540 Name=sam_seq-27 Failed, Run time 00:28:28, FAILED,
ExitCode 1
```

Ahora, dentro del directorio de output se crea un directorio llamado *slurm* donde se guardan todos los logs de ejecucion de las tareas. Hemos de buscar este fallo en particular por el *jobid*,

```
[osotolongo@brick03 WGS]$ ls slurm/ | grep 154540
bwa_seq-27-154540
```

una vez localizado el *log* correcto podeos buscar el error manualmente,

```
[osotolongo@brick03 WGS]$ less slurm/bwa_seq-27-154540
```

o simplemente hacer algun grep,

```
[osotolongo@brick03 WGS]$ grep -i error slurm/bwa_seq-27-154540
[gzread] Input/output error
```

Como informacion adicional, en el directorio slurm tambien se guardan los scripts a ejecutar para cada parte del proceso,

```
[osotolongo@brick03 WGS]$ ls slurm/ | grep seq-27 | grep sh
bwa_seq-27_0.sh
bwa_seq-27_1.sh
bwa_seq-27_2.sh
bwa_seq-27_3.sh
merge_seq-27.sh
validate_seq-27.sh
verify_seq-27.sh
```

El nombre del log nos dice que ha fallado alguno de los *bwa*. En caso de querer depurar mas lo que ha pasado podemos leer el log y encontrar cual de los comandos ha fallado exactamente.

ejemplo 2

Algo menos basico, recibimos un email con subject,

```
SLURM Job_id=154595 Name=validate_seq-8 Failed, Run time 01:01:18, FAILED,
ExitCode 2
```

y buscamos el log correspondiente,

```
[osotolongo@brick03 WGS]$ ls slurm/ | grep 154595
```

```
validate_seq-8-154595
```

Aqui si hacemos,

```
[osotolongo@brick03 WGS]$ grep -i error slurm/validate_seq-8-154595
```

obtenemos un monton de informacion ya que el error ocurrio a un nivel y todos los niveles posteriores arrojan errores. Asi que no hay mas remediio que leerse el log con calma, y vemos que mientras se hace **MarkDuplicates** tenemos un error de *Java*,

```
Exception in thread "main" picard.PicardException: Exception writing
ReadEnds to file.
        at
picard.sam.markduplicates.util.ReadEndsForMarkDuplicatesCodec.decode(ReadEnd
sForMarkDuplicatesCodec.java:110)
        at
picard.sam.markduplicates.util.ReadEndsForMarkDuplicatesCodec.decode(ReadEnd
sForMarkDuplicatesCodec.java:32)
        at
htsjdk.samtools.util.SortingCollection$FileRecordIterator.advance(SortingCol
lection.java:630)
        at
htsjdk.samtools.util.SortingCollection$FileRecordIterator.next(SortingCollec
tion.java:620)
        at htsjdk.samtools.util.PeekIterator.peek(PeekIterator.java:69)
        at
htsjdk.samtools.util.SortingCollection$PeekFileRecordIteratorComparator.comp
are(SortingCollection.java:655)
        at
htsjdk.samtools.util.SortingCollection$PeekFileRecordIteratorComparator.comp
are(SortingCollection.java:652)
        at java.util.TreeMap.compare(TreeMap.java:1295)
        at java.util.TreeMap.put(TreeMap.java:538)
        at java.util.TreeSet.add(TreeSet.java:255)
        at
htsjdk.samtools.util.SortingCollection$MergingIterator.next(SortingCollectio
n.java:566)
        at
picard.sam.markduplicates.MarkDuplicates.generateDuplicateIndexes(MarkDuplic
ates.java:729)
        at
picard.sam.markduplicates.MarkDuplicates.doWork(MarkDuplicates.java:259)
        at
picard.cmdline.CommandLineProgram.instanceMain(CommandLineProgram.java:305)
        at
picard.cmdline.PicardCommandLine.instanceMain(PicardCommandLine.java:103)
        at picard.cmdline.PicardCommandLine.main(PicardCommandLine.java:113)
Caused by: java.io.IOException: failed to read chunk
        at
org.xerial.snappy.SnappyInputStream.hasNextChunk(SnappyInputStream.java:433)
        at
```

```
org.xerial.snappy.SnappyInputStream.read(SnappyInputStream.java:167)
        at java.io.DataInputStream.readFully(DataInputStream.java:195)
        at java.io.DataInputStream.readLong(DataInputStream.java:416)
        at
picard.sam.markduplicates.util.ReadEndsForMarkDuplicatesCodec.decode(ReadEnd
sForMarkDuplicatesCodec.java:92)
        ... 15 more
```

Este error desencadena todos los demas errores.

Dado que sabemos que este log corresponde al script *validate_seq-8.sh* podemos encontrar rapidamente el comando culpable del error,

```
[osotolongo@brick03 WGS]$ grep MarkDuplicates slurm/validate_seq-8.sh
java -Djava.io.tmpdir=/nas/osotolongo/tmp/ -Xmx8g -jar
/nas/usr/local/bin/picard.jar MarkDuplicates
I=/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-8/tmp/seq8_sorted.bam
O=/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-8/tmp/seq8_GATKready.bam
METRICS_FILE=/the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-8/tmp/seq8_metrics.t
xt QUIET=true MAX_RECORDS_IN_RAM=2000000 ASSUME_SORTED=TRUE
CREATE_INDEX=TRUE
```

e intentar reparar lo que este mal.

Opciones extra

lista de inclusion

Digamos que de un pool de sujetos han fallado 2 o 3. No queremos tener que ejecutar el analisis para todos los sujetos, solo para los que han fallado. No queremos mover los sujetos que han fallado a un nuevo directorio pues atenta contra la organizacion del trabajo y el uso correcto del tiempo.

Para esto existe la opcion *-cut* que indica al script que se analice exclusivamente los sujetos contenidos en un archivo de input.

La forma correcta de utilizarlo es la siguiente. Ejemplo, si solo queremos que se ejecute el analisis en los sujetos *seq-8* y *seq-27*, hacemos el siguiente archivo,

```
[osotolongo@brick03 WGS]$ cat /nas/osotolongo/wgs/only.txt
seq-8
seq-27
```

y tras esto ejecutamos algo como,

```
$ ./wgs.py -cut /nas/osotolongo/wgs/only.txt -o
/the_dysk/BGI_exome/F18FTSEUET0180/WGS -s
/the_dysk/BGI_exome/F18FTSEUET0180/HUMehbE
```

que hara todo el procedimiento pero solo para estos sujetos.

debug

Para ayudar en la localizacion de errores existe la opcion *-g*, para indicar que no se borren los archivos temporales. El comando,

```
$ ./wgs.py -g -o /the_dysk/BGI_exome/F18FTSEUET0180/WGS -s
/the_dysk/BGI_exome/F18FTSEUET0180/HUMehbE
```

deja un subdirectorio *tmp* por cada sujeto con sus archivos temporales correspondientes. Ejemplo para seq-5, estos archivos estarian en */the_dysk/BGI_exome/F18FTSEUET0180/WGS/seq-5/tmp/* .