Project created on 04/17/2019 10:25.

Report for project African Centre for Gene Technologies (Dept. Bioinformatics and Computational Biology Unit)

Experiment created on 04/17/2019 17:11.

Analysis of germ line variants in cancer-related genes of black female South African breast cancer patients

No description

Task created on 04/19/2019 12:31.

Extracting raw reads from 166 samples

Due date: 04/18/2019 00:00 Completed on 04/19/2019 15:38

A total of 166 blood samples were previously collected from black South African females with breast carcinoma. The patients visited the "Oncology Clinic" at Steve Biko Hospital. Consent for all samples were given by patients together with etchics approval from KCT265 and 260/2018. DNA was obtained from the peripheral blood samples by the procedure illustrated by Johns and Paulus-Thomas et al. (1989). All samples were tested for the presence of BCRA mutations, however all of the samples tested negative. Samples were analyzed for germ line variants in cancer-related genes of black fenake South african breast cancer patients.

Task tags: Blood samples only (Germ line variants)

Completed by Junior MP on 04/19/2019 15:38.

Step 1: DNA sequencing Completed

DNA samples were sent to Omega Biotech in Georgia, USA, for sequencing using rhe Illumina TruSight Cancer Panel. The panel contained 94 carcinoma related genes and 284 SNPs previously identified to be associated with a susceptibility for carcinoma.

The commands that were used can be found under the section "Checklist" as in the previous step.

[Illumina_Inc__(2019)_Illumina-TruSight-Cancer-P...] File uploaded on 04/19/2019 13:50.

[Illumina_Inc_(2019)1-Seq-RUO-DNA-https_www_illu...] File uploaded on 04/19/2019 13:50.

Comments for step DNA sequencing

No comments

Completed by Junior MP on 04/19/2019 15:38.

Step 2: Copy OmegaFastQ files Completed

Sequencing files were copied to /nlustre/user/junior/OmegaFastq, which is the server where the big data is processed.

[Commands in Linux] Checklist created on 04/19/2019 15:38.

```
☑ mkdir OmegaFastq (Directory to save the files)
☑ cp -R /nlustre/users/fourie/OmegaFastq /nlustre/users/junior/OmegaFastq/ (Copy all fastq files from 166
patients from supervisor cluster to my directory)
☑ #!/usr/bin/env python
#Performs Fastqc on all the files and writes them in the OmegaFast directory
import sys
import os #create list in Python, which contains the command-line arguments passed to the script infile =
sys.argv[1] #Declaring variables
base_name = infile.replace('.fastq.gz', '')
outdir = base_name + '_fastqc'
script_name = 'run_fastqc_' + base_name + '.sh' #write to file
fh = open(script_name, 'w')
fh.write('#!/usr/bin/bash\n')
job_name = 'fastqc_' + base_name
fh.write('#PBS -N ' + job_name + '\n')
fh.write('#PBS -q long\n')
fh.write('#PBS -l walltime=00:15:00\n')
fh.write('#PBS -l nodes=1:ppn=28\n')
fh.write('#PBS -k oe\n')
fh.write('module load fastqc-0.11.7\n')
fh.write('cd /nlustre/users/junior/OmegaFastq\n')
fh.write('mkdir ' + outdir + '\n') #Perform Fastqc on the files in OmegaFastq directory
fastqc_command = 'fastqc -t 28 -o ' + outdir + ' ' + infile
fh.write(fastqc_command + '\n')
fh.close()
☑ cd /nlustre/users/junior/
mkdir FastQC ((to have one directory where Multiqc
can output these files )
☑ cd /nlustre/users/junior/OmegaFastq/
☑ module avail
multiqc --help
multiqc -i "Multiqc report of all the 166 samples" -o /nlustre/users/junior/FastQC/ .
☑ firefox Multiqc-report-of-all-the-166-samples_multiqc_report.html & (running the file on the background to
check the overal quality of all the samples)
```

Comments for step Copy OmegaFastQ files

No comments

Results after copying the sample files *[Results.png.pdf]* Uploaded by Junior MP on 04/19/2019 15:45.

Comments for result Results after copying the sample files

No comments

Samples of task Extracting raw reads from 166 samples

No items

Task created on 04/19/2019 15:49.

Non-GATK

Due date: 04/23/2019 00:00 Completed on 04/23/2019 15:27

The first step after performing the Quality analysis was trimming the samples. FastX_toolkit was used to trim 5 and 95 nucleotides on the 5' and 3' ends of the 100bp paired-end reads respectively.

Task tags: Pre-processing

Created by Junior MP on 04/23/2019 15:24.

Step 1: Inspecting MultiQC for all samples Uncompleted

MultiQC is a reporting tool that parses summary statistics from results and log files generated by other bioinformatics tools. MultiQC doesn't run other tools for you - it's designed to be placed at the end of analysis pipelines or to be run manually when you've finished running your tools. When is launched MultiQC, it recursively searches through any provided file paths and finds files that it recognises and it parses relevant information from these and generates a single stand-alone HTML report file. It also saves a directory of data files with all parsed data for further downstream use.
Ewels P. (21 Dec 2018) , *Using Multiqc.* Retrieved from https://multiqc.info/docs/

In the section "Files", the Multiqc of all the samples can be found.

Note: The first script that I wrote did not work on .gz files. In order to make use of the Fastx_toolkit you need to unzip the files first.
Thats why I created the second simple script to run it.

[Commands that were used for trimming the samples 5 and 95 nucleotides on the 5'and 3' ends of the 100bp paired-end reads]

Checklist created on 04/23/2019 15:07.

- [ ] #!/usr/bin/env python import sys
```
import os infile = sys.argv[1] trim_name = infile.replace('.fastq.gz', '')
outdir = trim_name + '_trimmed'
script_name = trim_name + '_trim_raw' + '.sh' fh = open(script_name, 'w')
fh.write('#!/usr/bin/bash\n')
job_name = 'FastX_'+ trim_name
fh.write('#PBS -N' + job_name + '\n')
fh.write('#PBS -q long\n')
fh.write('#PBS -l walltime=01:00:00\n')
fh.write('#PBS -l nodes=1:ppn=28\n')
fh.write('#PBS -k oe\n')
fh.write('module load fastx_toolkit-0.0.14\n')
fh.write('cd /nlustre/users/junior/OmegaFastq\n')
fh.write('mkdir ' + outdir + '\n')
fastx_command = 'fastx_trimmer -f 5 -l 95 -Q33 -i infile' + '-o outdir '
fh.write(fastx_command + '\n')
fh.close()
```
- [ ] for file in `ls |grep fastq` ; do fastx_trimmer -f l -l 95 -Q33 -i ${file} -o ${file}_trimmed; done

[FastQCMeanQaulityScores.jpeg] File uploaded on 04/23/2019 15:24.

FastQC: Mean Quality Scores

Comments for step Inspecting MultiQC for all samples

No comments

Samples of task Non-GATK

No items

Task created on 04/23/2019 15:28.

GATK

Due date: 04/30/2019 00:00 Completed on 04/30/2019 10:36

All samples that passed the qualitiy control were evaluated using the GATK best practices approach by the means of the BBCBIO pipeline which can be found in the references.
First the samples were mapped against the reference genome "hg19". For this step BWA MEM was used. Then SAMtools was used to view,sort and filter the alligned reads. Next, Qualimap was used to calculate how many reads there were alligned to each gene of interest.
Duplicate reads were marked using Picard, as was the base quality score recalibration. This step is needed to find systematic errors that were present or artefacts that could caused errors.
Variant calling was performed using the Haplotyper in gVCF. This HaplotypeCaller runs per-sample to generate an intermediate genomic gVCF (gVCF), which can then be used for joint genotyping of multiple samples in a very efficient way, which enables rapid incremental processing of samples as they roll off the sequencer, as well as scaling to very large cohort sizes.
In the last step specified cutoff-based filering of variants with VariantFiltration was used using the dfault filtering cut-offs.

Task tags: Germ line variant calling

Completed by Junior MP on 04/30/2019 10:36.

Step 1: BWA-mem mapping Completed

BWA: is a software package for mapping low-divergent sequences against a large reference genome, such as the human genome. It consists of three algorithms: BWA-backtrack, BWA-SW and BWA-MEM. The first algorithm is designed for Illumina sequence reads up to 100bp, while the rest two for longer sequences ranged from 70bp to 1Mbp. BWA-MEM and BWA-SW share similar features such as long-read support and split alignment, but BWA-MEM, which is the latest, is generally recommended for high-quality queries as it is faster and more accurate. BWA-MEM also has better performance than BWA-backtrack for 70-100bp Illumina reads.

[Commands used for mapping against hg19] Checklist created on 04/30/2019 10:36.

☐
```
#!/usr/bin/env python import sys
import os infile = sys.argv[1] sample_name = infile.replace('_R1_001.fastq','')
outdir = sample_name.replace('.fastq','')
script_name = outdir + '_bwa_mem' + '.sh' fh = open(script_name, 'w')
fh.write('#!/usr/bin/bash\n')
job_name = '\tAlligned_'+ sample_name
fh.write('#PBS -N' + job_name + '\n')
fh.write('#PBS -q long\n')
fh.write('#PBS -l walltime=01:30:00\n')
fh.write('#PBS -l nodes=1:ppn=28\n')
fh.write('#PBS -k oe\n')
fh.write('module load bwa-0.7.17\n')
fh.write('cd /nlustre/users/junior/OmegaFastq/fastq\n')
fh.write('mkdir ' + outdir + '\n') bwa_mem_command = 'bwa mem -t 28 -R \"@RG\\tID:' + sample_name +
'\\tSM:' + sample_name + '\\tPL:ILLUMINA\\tPU:' + sample_name + '\\tLB:' + sample_name + '\" ' +
'/nlustre/users/fourie/H.sapiens/gatk_resource_bundle/2.8/hg19/ucsc.hg19.fasta ' + sample_name +
```

'_R1_001.fastq ' + sample_name + '_R2_001.fastq > ' + sample_name + '.sam'
fh.write(bwa_mem_command + '\n')
fh.close()

- ☐ In this command the - R option stands for "read group " are added. These tags, when assigned appropriately, allow us to differentiate not only samples, but also various technical features that are associated with artifacts. With this information in hand, we can mitigate the effects of those artifacts during the duplicate marking and base recalibration steps.
- ☐ mkdir SAM (to move all the SAM files from the mapping in the directory)
- ☐ for file in `ls |grep sam`; do mv $file ./SAM; done (move all the SAM files to the directory)

Comments for step BWA-mem mapping

No comments

Samples of task GATK

No items

Task created on 04/30/2019 10:39.

GATK (SAM tools)

Due date: 05/02/2019 00:00 Completed on 05/02/2019 10:14

Samtools is a set of utilities that manipulate alignments in the SAM/BAM format. It imports from and exports to the SAM (Sequence Alignment/Map) format, does sorting, merging and indexing, and allows to retrieve reads in any regions swiftly.

Samtools is designed to work on a stream. It regards an input file `-' as the standard input (stdin) and an output file `-' as the standard output (stdout). Several commands can thus be combined with Unix pipes. Samtools always output warning and error messages to the standard error output (stderr).

Task tags: SAM

Completed by Junior MP on 04/30/2019 14:40.

Step 1: SAM tools/Qualimap/ MultiQC Completed

In the first step **samtool view/import** was used. With no options or regions specified, it prints all alignments in the specified input alignment file (in SAM, BAM, or CRAM format) to standard output in SAM format (with no header). The analyst may specify one or more space-separated region specifications after the input filename to restrict output to only those alignments which overlap the specified region(s). Use of region specifications requires a coordinate-sorted and indexed input file (in BAM or CRAM format).

samtools view [*options*] *in.sam*|*in.bam*|*in.cram* [*region*...]

Next, the **samtools sort** command was used to sort alignments by leftmost coordinates, or by read name when **-n** is used. An appropriate sort order header tag will be added or an existing one updated if necessary. The sorted output is written to standard output by default, or to the specified file (*out.bam*) when **-o** is used.

samtools sort [**-l** *level*] [**-m** *maxMem*] [**-o** *out.bam*] [**-O** *format*] [**-n**] [**-t** *tag*] [**-T** *tmpprefix*] [**-@** *threads*] [*in.sam*|*in.bam*|*in.cram*]

Then **samtool index** was used to index a coordinate-sorted BAM or CRAM file for fast random access. However, this option does not work with SAM files even if they are bgzip compressed — to index such files, the analyst must use tabix instead.

samtools index [**-bc**] [**-m** *INT*] *aln.bam*|*aln.cram* [*out.index*]

Taken from : http://www.htslib.org/doc/samtools.html
More information can be found on http://www.htslib.org/doc/samtools.html

Note: The only picture that could be taken was of one sample SAM file after mapping with BWA-MEM. The BAM files are binary files so it was not possible to read the files in command-line.

The next step consists of using the tool **Qualimap** (-BamQC). With a given BAM file and an annotation (GTF/GFF or BED file), this tool calculates how many reads are mapped to each region of interest.

qualimap bamqc -bam file.bam -outdir qualimap_results ==> The only problem with Qualimap is that you have to specify every output directory. It does not use one output directory to put in the results. The first analysis was run without this option, this made all of the files overwrite eachother. In the next analysis every sample had his/her own output directory.

Taken from : http://qualimap.bioinfo.cipf.es/doc_html/command_line.html
More information can be found on http://qualimap.bioinfo.cipf.es/doc_html/command_line.html

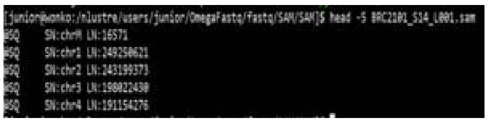
In the last step a multiqc is ran on all the bam files to inspect the qaulity of the mapped,indexed and sorted reads.

[SAM tools commands]   Checklist created on 04/30/2019 11:53.

☐ mkdir SAM (directory to have all the SAM files after mapping)
☐ for file in `ls |grep sam`; do mv $file ./SAM; done (move the files to the directory)
cd SAM/
nano SAM-to-BAM.py (create Python script)
☐ #!/usr/bin/env python import sys
import os infile = sys.argv[1] sample_name = infile.replace('.sam', '')
script_name = sample_name + '_bam' + '.sh' fh = open(script_name, 'w')
fh.write('#!/usr/bin/bash\n')
job_name = '\tBam_'+ sample_name
fh.write('#PBS -N' + job_name + '\n')
fh.write('#PBS -q long\n')
fh.write('#PBS -l walltime=01:30:00\n')
fh.write('#PBS -l nodes=1:ppn=1\n')
fh.write('#PBS -k oe\n')
fh.write('module load samtools-1.7\n')
fh.write('cd /nlustre/users/junior/OmegaFastq/fastq/SAM\n')
bam_view_command = 'samtools view -bt
/nlustre/users/fourie/H.sapiens/gatk_resource_bundle/2.8/hg19/ucsc.hg19.fasta -o ' + sample_name +'.bam
' + sample_name + '.sam'
fh.write(bam_view_command + '\n')
fh.close()
☐ for file in `ls |grep sam`; do ./SAM-to-BAM.py $file; done (creates the bash scripts for Torque)
☐ for file in `ls |grep sh`; do qsub $file; done (send the scripts to the Torque environment to be executed)
☐ cd /nlustre/users/junior/OmegaFastq/fastq/SAM/
mkdir BAM
for file in `ls |grep bam`; do mv $file ./BAM/;done (move all the BAM files to this directory)
cd /nlustre/users/junior/OmegaFastq/fastq/SAM/
mkdir Scripts
for file in `ls |grep sh`; do mv $file ../Scripts/;done (move all the scripts to this directory = data clean-up)
☐ cd BAM/
nano SAM-sort.py (create the Python script to sort)
☐ #!/usr/bin/env python import sys
import os infile = sys.argv[1] sample_name = infile.replace('.bam', '')
script_name = sample_name + '_sort' + '.sh' fh = open(script_name, 'w')
fh.write('#!/usr/bin/bash\n')
job_name = '\tSort_'+ sample_name
fh.write('#PBS -N' + job_name + '\n')
fh.write('#PBS -q long\n')
fh.write('#PBS -l walltime=01:30:00\n')
fh.write('#PBS -l nodes=1:ppn=4\n')
fh.write('#PBS -k oe\n')
fh.write('module load samtools-1.7\n')
fh.write('cd /nlustre/users/junior/OmegaFastq/fastq/SAM/BAM\n')
bam_sort_command = 'samtools sort -o ' + sample_name +'.sorted.bam ' + sample_name + '.bam'
fh.write(bam_sort_command + '\n')
fh.close()
☐ chmod +755 SAM-sort.py
☐ for file in `ls |grep bam`; do ./SAM-sort.py $file;done
☐ for file in `ls | grep sh`; do qsub $file;done
☐ mkdir Scripts

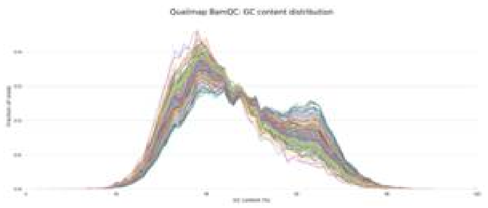
```
for file in `ls |grep sh`; do mv $file ./Scripts/ ; done (data clean-up)
mkdir Sorted
for file in `ls |grep sorted`; do mv $file ./Sorted/ ; done
☐ cd Sorted/
nano SAM-index.py (create the Python script to sort)
☐ #!/usr/bin/env python import sys
import os infile = sys.argv[1] sample_name = infile.replace('.bam', '')
script_name = sample_name + '_index' + '.sh' fh = open(script_name, 'w')
fh.write('#!/usr/bin/bash\n')
job_name = '\tIndex_'+ sample_name
fh.write('#PBS -N' + job_name + '\n')
fh.write('#PBS -q long\n')
fh.write('#PBS -l walltime=01:30:00\n')
fh.write('#PBS -l nodes=1:ppn=4\n')
fh.write('#PBS -k oe\n')
fh.write('module load samtools-1.7\n')
fh.write('cd /nlustre/users/junior/OmegaFastq/fastq/SAM/BAM\n')
bam_index_command = 'samtools index -b ' + sample_name +'.sorted.bam '
fh.write(bam_index_command + '\n')
fh.close()
☐ chmod +755 SAM-index.py
☐ for file in `ls | grep bam`; do ./SAM-index.py $file;done
☐ for file in `ls|grep sh`; do qsub $file;done
for file in `ls|grep sh`; do mv $file ./Scripts/;done (data clean-up)
☐ cd /nlustre/users/junior/OmegaFastq/fastq/SAM/BAM/Sorted/
nano BAMQC.py
☐ #!/usr/bin/env python import sys
import os infile = sys.argv[1] sample_name = infile.replace('.sorted.bam', '')
outdir = sample_name
script_name = sample_name + '_qaulimap' + '.sh' fh = open(script_name, 'w')
fh.write('#!/usr/bin/bash\n')
job_name = '\tBamqc_'+ sample_name
fh.write('#PBS -N' + job_name + '\n')
fh.write('#PBS -q long\n')
fh.write('#PBS -l walltime=01:30:00\n')
fh.write('#PBS -l nodes=1:ppn=8\n')
fh.write('#PBS -k oe\n')
fh.write('module load qualimap-2.2.1\n')
fh.write('cd /nlustre/users/junior/OmegaFastq/fastq/SAM/BAM/Sorted\n')
fh.write('mkdir ' + outdir + '\n' )
bam_index_command = 'qualimap bamqc -bam ' + sample_name +'.sorted.bam -gff
/nlustre/users/fourie/BIFHons/Mapping/ucsc_hg19_refseq.gtf -outdir ' + outdir
fh.write(bam_index_command + '\n')
fh.close()
☐ chmod +755 BAMQC.py
☐ for file in `ls|grep sorted`; do ./BAMQC.py $file;done
for file in `ls |grep sh`; do qsub $file; done
☐ module load multiqc
multiqc . (searches for files in current directory to produce MultiQC report)
```

[SAM_file_after_mapping_with_BWA-MEM_.png] File uploaded on 04/30/2019 11:19.



```
[junior@wonko:/nlustre/users/junior/OmegaFastq/fastq/SAM/SAM]$ head -5 BRC2101_S14_L001.sam
@SQ     SN:chrM LN:16571
@SQ     SN:chr1 LN:249250621
@SQ     SN:chr2 LN:243199373
@SQ     SN:chr3 LN:198022430
@SQ     SN:chr4 LN:191154276
```

[BamQC_GC_content.png] File uploaded on 04/30/2019 14:38.



Comments for step SAM tools/Qualimap/ MultiQC

No comments

Samples of task GATK (SAM tools)

No items

Task created on 04/30/2019 14:40.

GATK (Picard)

Due date: 05/02/2019 00:00 Completed on 05/02/2019 14:31

This tool locates and tags duplicate reads in a BAM or SAM file, where duplicate reads are defined as originating from a single fragment of DNA. Duplicates can arise during sample preparation e.g. library construction using PCR.

Task tags: Marking duplicates

Completed by Junior MP on 05/03/2019 08:59.

Step 1: Duplicates marking Completed

Almost all statistical models for variant calling assume some sort of independence between measurements. The duplicates (if one assumes that they arise from PCR artifact) are not independent. This lack of independence will usually lead to a breakdown of the statistical model and measures of statistical significance that are incorrect.

There are experiments where one should not make the assumption that reads that have the same start positions are PCR duplicates. In that case, using MarkDuplicates is not justified.

[Commands for Duplicate marking] Checklist created on 05/02/2019 10:21.

- ☐ cd OmegaFastq/fastq/SAM/BAM/Sorted/ (or to the directory where the sorted bam files can be found)
- ☐ nano Picard.py (create a Python file)
- ☐ #!/usr/bin/env python import sys

```
import os infile = sys.argv[1] sample_name = infile.replace('.sorted.bam', '')
script_name = sample_name + '_markedDup' + '.sh' fh = open(script_name, 'w')
fh.write('#!/usr/bin/bash\n')
job_name = '\tMarkedDup_'+ sample_name
fh.write('#PBS -N' + job_name + '\n')
fh.write('#PBS -q long\n')
fh.write('#PBS -l walltime=01:30:00\n')
fh.write('#PBS -l nodes=1:ppn=28\n')
fh.write('#PBS -k oe\n')
fh.write('module load picard-2.17.11\n')
fh.write('cd /nlustre/users/junior/OmegaFastq/fastq/SAM/BAM/Sorted\n')
mark_dup_command = 'java -jar $PICARD MarkDuplicates I='+ sample_name + '.sorted.bam O='+
sample_name +'_dup.bam M='+ sample_name + '_dup.metrics TAGGING_POLICY=All'
fh.write(mark_dup_command + '\n')
fh.close()
```

Comments for step Duplicates marking

No comments

Samples of task GATK (Picard)

No items

Task created on 05/03/2019 09:00.

GATK (BQRS)

Due date: 05/07/2019 00:00

Generates recalibration table for Base Quality Score Recalibration (BQSR)

Task tags: Base Quality Recalibration Scores

Created by Junior MP on 05/03/2019 14:32.

Step 1: Base Quality Score Recalibrator Uncompleted

First step of the two-step of the base quality score recalibration process is completed with the gatk-tool "BQSR". This tool generates a recalibration table based on various covariates. The default covariates are read group, reported quality score, machine cycle, and nucleotide context. The BQRS generates tables based on specified covariates. It does a by-locus traversal operating only at sites that are in the known sites VCF. ExAc, gnomAD, or dbSNP resources can be used as known sites of variation. We assume that all reference mismatches we see are therefore errors and indicative of poor base quality. Since there is a large amount of data one can then calculate an empirical probability of error given the particular covariates seen at this site, where p(error) = num mismatches / num observations. The output file is a table (of the several covariate values, num observations, num mismatches, empirical quality score).

More info
on: https://software.broadinstitute.org/gatk/documentation/tooldocs/4.0.6.0/org_broadinstitute_hellbender_tools_walkers_bqsr_BaseRecalibrator.php
Taken
from: https://software.broadinstitute.org/gatk/documentation/tooldocs/4.0.6.0/org_broadinstitute_hellbender_tools_walkers_bqsr_BaseRecalibrator.php

The second step of the base quality score recalibration process is processed with the tool "PrintReads GATK3)/ ApplyBQRS (GATK4)" and is used to apply base quality score recalibration.

This tool performs the second pass in a two-stage process called Base Quality Score Recalibration (BQSR). Specifically, it recalibrates the base qualities of the input reads based on the recalibration table produced by the BaseRecalibrator tool, and outputs a recalibrated BAM or CRAM file.

[BaseRecalibrator commands] Checklist created on 05/03/2019 14:32.

- ☐ mkdir Scripts Metrics Sortedbams (clean-up the files system)
nano BQSR.py
chmod +755 BQSR.py
- ☐ #!/usr/bin/env python import sys
import os infile = sys.argv[1] sample_name = infile.replace('_dup.bam', '')
script_name = sample_name + '_BQSR1' + '.sh' fh = open(script_name, 'w')
fh.write('#!/usr/bin/bash\n')
job_name = '\tBQSR_'+ sample_name
fh.write('#PBS -N' + job_name + '\n')
fh.write('#PBS -q long\n')
fh.write('#PBS -l walltime=01:30:00\n')
fh.write('#PBS -l nodes=1:ppn=24\n')
fh.write('#PBS -k oe\n')
fh.write('module load gatk-4.0.4.0\n')
fh.write('cd /nlustre/users/junior/OmegaFastq/fastq/SAM/BAM/Sorted\n')
BQRS_command = 'gatk BaseRecalibrator -I '+ sample_name + '_dup.bam -R /nlustre/users/fourie/H.sapiens/gatk_resource_bundle/2.8/hg19/ucsc.hg19.fasta --known-sites /nlustre/users/fourie/H.sapiens/gatk_resource_bundle/2.8/hg19/dbsnp_138.hg19.vcf --known-sites /nlustre/users/fourie/H.sapiens/gatk_resource_bundle/2.8/hg19/1000G_phase1.indels.hg19.vcf --known-sites /nlustre/users/fourie/H.sapiens/gatk_resource_bundle/2.8/hg19/1000G_phase1.snps.high_confidence.hg19.vcf --known-sites /nlustre/users/fourie/H.sapiens/gatk_resource_bundle/2.8/hg19/Mills_and_1000G_gold_standard.indels.hg19.vcf -O '+ sample_name +'_data.table'
fh.write(BQRS_command + '\n')
fh.close()
- ☐ qsub all the files
- ☐ Second part (ApplyBQSR)
- ☐ for file in `ls|grep sh`; do mv ./Scripts; done (clean-up)
- ☐ nano ApplyBQSR.py
chmod +755 ApplyBQSR.py
- ☐ #!/usr/bin/env python import sys

```
import os infile = sys.argv[1] sample_name = infile.replace('_dup.bam', '')
script_name = sample_name + '_ApplyBQSR' + '.sh' fh = open(script_name, 'w')
fh.write('#!/usr/bin/bash\n')
job_name = '\tApplyBQSR_'+ sample_name
fh.write('#PBS -N' + job_name + '\n')
fh.write('#PBS -q long\n')
fh.write('#PBS -l walltime=01:30:00\n')
fh.write('#PBS -l nodes=1:ppn=24\n')
fh.write('#PBS -k oe\n')
fh.write('module load gatk-4.0.4.0\n')
fh.write('cd /nlustre/users/junior/OmegaFastq/fastq/SAM/BAM/Sorted\n')
ApplyBQRS_command = 'gatk ApplyBQSR -R
/nlustre/users/fourie/H.sapiens/gatk_resource_bundle/2.8/hg19/ucsc.hg19.fasta -I '+ sample_name +
'_dup.bam --bqsr-recal-file '+ sample_name +'_data.table -O '+ sample_name +'_Recali.bam'
fh.write(ApplyBQRS_command + '\n')
fh.close()
```

Comments for step Base Quality Score Recalibrator

No comments

Samples of task GATK (BQRS)

No items