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 with specific effect prediction of non-coding 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 female South African breast cancer patients.

Task tags: Blood samples only ( Germ line variants)

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

*No protocol description*

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 the 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

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

*No protocol description*

Created by Junior MP on 05/09/2019 08:55.

**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 MultiQC is launched, 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.

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 quality control were later 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. SAMtools was used next, in order 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. The tool also provided 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. Moreover, this tool can split SNVs and indels.
In the last step specified cutoff-based filtering of variants with VariantFiltration was used using the default filtering cut-offs.

Task tags: Germ line variant calling

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

*No protocol description*

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 which manipulates alignments in SAM/BAM format. It imports from and exports to the SAM (Sequence Alignment/Map) format, does sorting, merging and indexing, and allows researchers to retrieve reads in any regions swiftly.

Samtools is designed to work on a various pipelines. 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 pipelines. Samtools always output warning and error messages to the standard error output (stderr).

Task tags: SAM

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

---

*No protocol description*

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.

*[ 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

Picard 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

Protocol created on 04/30/2019

14:40.

*No protocol description*

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)
- [x] #!/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 (BQSR)

Due date: 05/07/2019 00:00 Completed on 05/07/2019 08:08

Generates recalibration table for Base Quality Score Recalibration (BQSR)

Task tags: Base Quality Score Recalibration

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

*No protocol description*

Completed by Junior MP on 05/07/2019 08:08.

**Step 1:** Base Quality Score Recalibrator
**Completed**

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_confid
```

ence.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 (BQSR)

*No items*

Task created on 05/07/2019 08:09.

---

## GATK(HaplotypeCaller)

Due date: 05/10/2019 00:00 Completed on 05/10/2019 09:21

After recalibrating and applying the base quality scores, the next step is to process to variant calling. To complete this step HaplotypeCaller in gVCF mode will be used. The actual variant calling will consists of more than one step. In the protocol steps the other steps will be explained completely.

Task tags: Variantcalling (gVCF mode)

Protocol created on 05/07/2019 08:09.

---

*No protocol description*

Completed by Junior MP on 05/10/2019 09:21.

---

**Step 1:** HaplotypeCaller / GenotypeGVCFs **Completed**

The interesting part about variant calling is that now researchers are able to process,call gene differences in genomes. These variants can be SNP's and indels which can seen in VCF (variant call formats/files).

VCF is the standard file format for storing variation data. It is used by large scale variant mapping projects such as germ line variant discovery. It is also the standard output of variant calling software such as Genome Analysis Tool Kit (GATK) and the standard input for variant analysis tools such as the Variant Effect Predictor (VEP) or for variation archives like  European Variation Archive (EVA). VCF is a preferred format because it is unambiguous, scalable and flexible, allowing extra information to be added to the info field. Many millions of variants can be stored in a single VCF file.

More information can be found on : https://www.ebi.ac.uk/training/online/course/human-genetic-variation-introduction/exercise-title/want-know-how-we-did-it
Taken from :https://www.ebi.ac.uk/training/online/course/human-genetic-variation-introduction/exercise-title/want-know-how-we-did-it

Another format often used in variant calling is the gVCF. In short gVCF stands for Genomic VCF and  is a kind of VCF, so the basic format specification is the same as for a regular VCF, but a Genomic VCF contains extra information that can be used in downstream analysis.
One main difference between a VCF and gVCF would be that the gVCF has records for all sites, whether there is a variant call there or not (in the picture this is documented by the variants sites in red and the non variant sites in blue). The goal is to have every site represented in the file in order to do joint analysis of a cohort in subsequent steps. The records in a gVCF include an accurate estimation of how confident we are in the determination that the sites are homozygous-reference or not. This estimation is generated by the HaplotypeCaller's built-in reference model.

More information can be found on : https://sites.google.com/site/gvcftools/home/about-gvcf
Taken from :https://sites.google.com/site/gvcftools/home/about-gvcf

Performing variant calling consist of two steps: first the tool HaplotypeCaller is used to identify potential variants in each sample and makes intermediary gVCF (which will not be used is final analysis). If working with a large number of samples  variant calling can consist of three steps instead of two. All samples can be combined in one g.vcf.gz file using the tool "CombineVCFs" or "GenomicsDBImport".
In the last step, GATK's tool GenotypeGVCFs is used to perform join genotyping on the cohort variants.

More information can be found on: https://software.broadinstitute.org/gatk/documentation/tooldocs/4.0.4.0/org_broadinstitute_hellbender_tools_walkers_haplotypecaller_HaplotypeCaller.php#--emit-ref-confidence
and https://software.broadinstitute.org/gatk/documentation/tooldocs/4.0.4.0/org_broadinstitute_hellbender_tools_walkers_GenotypeGVCFs.php
Taken form: https://software.broadinstitute.org/gatk/documentation/tooldocs/4.0.4.0/org_broadinstitute_hellbender_tools_walkers_GenotypeGVCFs.php and https://software.broadinstitute.org/gatk/documentation/tooldocs/4.0.4.0/org_broadinstitute_hellbender_tools_walkers_haplotypecaller_HaplotypeCaller.php#--emit-ref-confidence

**Anything about VQSR?**

[ Variant calling commands ] Checklist created on 05/07/2019 09:59.

- [ ] mkdir Recali , Scripts , Tables and Dups
for file in `ls|grep Recali`; do mv $file ./Recali/;done
for file in `ls|grep sh`; do mv $file ./Scripts/;done
for file in `ls|grep table`; do mv $file ./Tables/;done
for file in `ls|grep dup.bam`;do mv $file ./Dups/;done ==> (data clean-up)
- [ ] cd Recali
nano HaplotypeCall.py (call the variants first)
- [ ] #!/usr/bin/env python import sys
import os infile = sys.argv[1] sample_name = infile.replace('_Recali.bam', '')
script_name = sample_name + '_HaploCall' + '.sh' fh = open(script_name, 'w')
fh.write('#!/usr/bin/bash\n')
job_name = '\t_HaploCall_'+ 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/Recali\n')
HaploCall_command = 'gatk HaplotypeCaller -R
/nlustre/users/fourie/H.sapiens/gatk_resource_bundle/2.8/hg19/ucsc.hg19.fasta -I '+
sample_name + '_Recali.bam -L
/nlustre/users/fourie/H.sapiens/intervals/trusight_cancer_manifest_a.bed -O '+ sample_name
+'.g.vcf.gz -ERC GVCF'
fh.write(HaploCall_command + '\n')
fh.close()
- [ ] chmod +755 HaplotypCall.py
- [ ] for file in `ls|grep Recali.bam`; do ./HaplotypCall $file;done
- [ ] for file in `ls|grep sh`; do qsub $file;done
- [ ] Second step CombineGVCFs
- [ ] cd Recali
- [ ] mkdir BamandBai, Scripts and TBIs
for file in `ls|grep tbi`; do mv $file ./TBIs/;done
for file in `ls|grep sh`; do mv $file ./Scripts/;done
for file in `ls|grep bam`;do mv $file ./BamandBai/;done
for file in `ls|grep bai`;do mv $file ./BamandBai/;done ==> (data clean-up)
- [ ] #!/usr/bin/env python import sys
import os infile = sys.argv[1] sample_name = infile.replace('.g.vcf.gz', '')
script_name = sample_name + '_Combine' + '.sh' fh = open(script_name, 'w')
fh.write('#!/usr/bin/bash\n')
job_name = '\t_Combine_'+ 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/Recali\n')
GenomicDB_command = 'gatk CombineGVCFs -R
/nlustre/users/fourie/H.sapiens/gatk_resource_bundle/2.8/hg19/ucsc.hg19.fasta -V
BRB101_S20_L001.g.vcf.gz -V BRB102_S24_L001.g.vcf.gz -V BRB104_S21_L001.g.vcf.gz -V
BRB106_S19_L001.g.vcf.gz -V BRB108_S1_L001.g.vcf.gz -V BRB10_S11_L001.g.vcf.gz -V
BRB111_S2_L001.g.vcf.gz -V BRB113_S3_L001.g.vcf.gz -V BRB114_S22_L001.g.vcf.gz -V
BRB118_S20_L001.g.vcf.gz -V BRB11_S12_L001.g.vcf.gz -V BRB120_S23_L001.g.vcf.gz -V
BRB121_S24_L001.g.vcf.gz -V BRB122_S4_L001.g.vcf.gz -V BRB123_S5_L001.g.vcf.gz -V
BRB124_S21_L001.g.vcf.gz -V BRB125_S22_L001.g.vcf.gz -V BRB129_S6_L001.g.vcf.gz -V
BRB130_S16_L001.g.vcf.gz -V BRB131_S17_L001.g.vcf.gz -V BRB132_S7_L001.g.vcf.gz -V
BRB137_S1_L001.g.vcf.gz -V BRB138_S2_L001.g.vcf.gz -V BRB139_S3_L001.g.vcf.gz -V
BRB142_S4_L001.g.vcf.gz -V BRB143_S8_L001.g.vcf.gz -V BRB146_S5_L001.g.vcf.gz -V
BRB147_S6_L001.g.vcf.gz -V BRB148_S9_L001.g.vcf.gz -V BRB14_S3_L001.g.vcf.gz -V
BRB150_S18_L001.g.vcf.gz -V BRB152_S10_L001.g.vcf.gz -V BRB153_S23_L001.g.vcf.gz -V
BRB154_S7_L001.g.vcf.gz -V BRB156_S8_L001.g.vcf.gz -V BRB158_S9_L001.g.vcf.gz -V

BRB160_S11_L001.g.vcf.gz -V BRB161_S24_L001.g.vcf.gz -V BRB162_S12_L001.g.vcf.gz -V
BRB166_S1_L001.g.vcf.gz -V BRB167_S19_L001.g.vcf.gz -V BRB169_S2_L001.g.vcf.gz -V
BRB170_S13_L001.g.vcf.gz -V BRB171_S3_L001.g.vcf.gz -V BRB172_S4_L001.g.vcf.gz -V
BRB173_S5_L001.g.vcf.gz -V BRB174_S6_L001.g.vcf.gz -V BRB175_S7_L001.g.vcf.gz -V
BRB177_S8_L001.g.vcf.gz -V BRB17_S15_L001.g.vcf.gz -V BRB182_S14_L001.g.vcf.gz -V
BRB185_S9_L001.g.vcf.gz -V BRB186_S10_L001.g.vcf.gz -V BRB187_S15_L001.g.vcf.gz -V
BRB188_S16_L001.g.vcf.gz -V BRB189_S17_L001.g.vcf.gz -V BRB18_S4_L001.g.vcf.gz -V
BRB190_S11_L001.g.vcf.gz -V BRB191_S12_L001.g.vcf.gz -V BRB193_S18_L001.g.vcf.gz -V
BRB194_S19_L001.g.vcf.gz -V BRB197_S10_L001.g.vcf.gz -V BRB199_S20_L001.g.vcf.gz -V
BRB19_S5_L001.g.vcf.gz -V BRB1_S9_L001.g.vcf.gz -V BRB200_S20_L001.g.vcf.gz -V
BRB201_S21_L001.g.vcf.gz -V BRB203_S11_L001.g.vcf.gz -V BRB205_S22_L001.g.vcf.gz -V
BRB207_S12_L001.g.vcf.gz -V BRB208_S21_L001.g.vcf.gz -V BRB20_S6_L001.g.vcf.gz -V
BRB215_S23_L001.g.vcf.gz -V BRB21_S7_L001.g.vcf.gz -V BRB220_S13_L001.g.vcf.gz -V
BRB224_S14_L001.g.vcf.gz -V BRB225_S15_L001.g.vcf.gz -V BRB226_S16_L001.g.vcf.gz -V
BRB229_S17_L001.g.vcf.gz -V BRB233_S1_L001.g.vcf.gz -V BRB234_S2_L001.g.vcf.gz -V
BRB236_S13_L001.g.vcf.gz -V BRB237_S18_L001.g.vcf.gz -V BRB238_S22_L001.g.vcf.gz -V
BRB239_S14_L001.g.vcf.gz -V BRB240_S15_L001.g.vcf.gz -V BRB241_S3_L001.g.vcf.gz -V
BRB242_S16_L001.g.vcf.gz -V BRB246_S4_L001.g.vcf.gz -V BRB247_S17_L001.g.vcf.gz -V
BRB249_S19_L001.g.vcf.gz -V BRB252_S5_L001.g.vcf.gz -V BRB253_S18_L001.g.vcf.gz -V
BRB254_S19_L001.g.vcf.gz -V BRB255_S20_L001.g.vcf.gz -V BRB257_S21_L001.g.vcf.gz -V
BRB258_S20_L001.g.vcf.gz -V BRB259_S21_L001.g.vcf.gz -V BRB260_S6_L001.g.vcf.gz -V
BRB261_S22_L001.g.vcf.gz -V BRB264_S7_L001.g.vcf.gz -V BRB265_S8_L001.g.vcf.gz -V
BRB268_S22_L001.g.vcf.gz -V BRB270_S9_L001.g.vcf.gz -V BRB271_S10_L001.g.vcf.gz -V
BRB272_S11_L001.g.vcf.gz -V BRB273_S23_L001.g.vcf.gz -V BRB275_S24_L001.g.vcf.gz -V
BRB276_S1_L001.g.vcf.gz -V BRB279_S12_L001.g.vcf.gz -V BRB281_S2_L001.g.vcf.gz -V
BRB282_S3_L001.g.vcf.gz -V BRB283_S4_L001.g.vcf.gz -V BRB284_S23_L001.g.vcf.gz -V
BRB286_S5_L001.g.vcf.gz -V BRB287_S6_L001.g.vcf.gz -V BRB288_S7_L001.g.vcf.gz -V
BRB28_S8_L001.g.vcf.gz -V BRB290_S24_L001.g.vcf.gz -V BRB2_S1_L001.g.vcf.gz -V
BRB34_S13_L001.g.vcf.gz -V BRB37_S9_L001.g.vcf.gz -V BRB38_S4_L001.g.vcf.gz -V
BRB39_S5_L001.g.vcf.gz -V BRB3_S1_L001.g.vcf.gz -V BRB42_S6_L001.g.vcf.gz -V
BRB44_S10_L001.g.vcf.gz -V BRB46_S7_L001.g.vcf.gz -V BRB47_S8_L001.g.vcf.gz -V
BRB48_S14_L001.g.vcf.gz -V BRB49_S15_L001.g.vcf.gz -V BRB50_S16_L001.g.vcf.gz -V
BRB51_S9_L001.g.vcf.gz -V BRB52_S10_L001.g.vcf.gz -V BRB53_S17_L001.g.vcf.gz -V
BRB55_S11_L001.g.vcf.gz -V BRB57_S12_L001.g.vcf.gz -V BRB58_S13_L001.g.vcf.gz -V
BRB59_S14_L001.g.vcf.gz -V BRB5_S2_L001.g.vcf.gz -V BRB62_S18_L001.g.vcf.gz -V
BRB68_S19_L001.g.vcf.gz -V BRB6_S10_L001.g.vcf.gz -V BRB70_S20_L001.g.vcf.gz -V
BRB72_S11_L001.g.vcf.gz -V BRB73_S15_L001.g.vcf.gz -V BRB74_S12_L001.g.vcf.gz -V
BRB75_S13_L001.g.vcf.gz -V BRB77_S14_L001.g.vcf.gz -V BRB78_S16_L001.g.vcf.gz -V
BRB81_S15_L001.g.vcf.gz -V BRB83_S16_L001.g.vcf.gz -V BRB84_S21_L001.g.vcf.gz -V
BRB87_S17_L001.g.vcf.gz -V BRB88_S22_L001.g.vcf.gz -V BRB89_S18_L001.g.vcf.gz -V
BRB8_S2_L001.g.vcf.gz -V BRB91_S19_L001.g.vcf.gz -V BRB94_S17_L001.g.vcf.gz -V
BRB96_S18_L001.g.vcf.gz -V BRB98_S23_L001.g.vcf.gz -V BRB99_S8_L001.g.vcf.gz -V
BRB9_S3_L001.g.vcf.gz -V BRC1341_S23_L001.g.vcf.gz -V BRC2101_S14_L001.g.vcf.gz -V
OVB1_S13_L001.g.vcf.gz -O Together.g.vcf.gz'
fh.write(GenomicDB_command + '\n')
fh.close()

- [ ] qsub BRB101_S20_L001_Combine.sh

Important note: The script did not work in the first place. This was due to the fact that I moved all the index files (.tbi-files) to another directory.
In order to avoid this problem/mistake perform the data clean-up step after GenotypeVCFs.

- [ ] Last step GenotypeGVCFs
- [ ] cd Recalli
- [ ] #!/usr/bin/env python import sys

import os infile = sys.argv[1] sample_name = infile.replace('.g.vcf.gz', '')
script_name = sample_name + '_GenotypeGVCFs' + '.sh' fh = open(script_name, 'w')
fh.write('#!/usr/bin/bash\n')
job_name = '\t_GenotypeGVCFs_'+ 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/Recali\n')
GenotypeGVCFs_command = 'gatk GenotypeGVCFs -R

```
/nlustre/users/fourie/H.sapiens/gatk_resource_bundle/2.8/hg19/ucsc.hg19.fasta -V
Together.g.vcf.gz -O GenotypeVCFs.vcf.gz'
fh.write(GenotypeGVCFs_command + '\n')
fh.close()
```

☐ qsub BRB101_S20_L001._GenotypeGVCFs.sh

*[ Example-of-VCF-_Paco._H_(2019)-High_Throughput_... ]* File uploaded on 05/07/2019 09:50.

*[ gvcf.png ]* File uploaded on 05/07/2019 09:50.

Comments for step HaplotypeCaller / GenotypeGVCFs

*No comments*

Samples of task GATK(HaplotypeCaller)

*No items*

Task created on 05/10/2019 14:35.

## Raw variants

Due date: 05/13/2019 00:00 Completed on 05/13/2019 15:14

After variant calling using the haplotypeCaller, variants were filtered using a specified cut-offs. This step consist of two tools, first we wil select the type of variants with the tool "SelectVariants". The choice wil be between SNP's and indels. After selecting the variants, "VariantFiltration"will be used to filter variant calls based on INFO and/or FORMAT annotations.

Task tags: Selection and filtering of variants

Protocol created on 05/10/2019 14:35.

*No protocol description*

Completed by Junior MP on 05/14/2019 13:18.

**Step 1:** SelectVariants & VariantFiltration

**Completed**

SELECTVARIANTS

Select a subset of variants from a VCF file, extract the Indels /SNPS's from the call set.

This tool makes it possible to select a subset of variants based on various criteria in order to facilitate certain analyses. Examples of such analyses include comparing and contrasting cases vs. controls, extracting variant or non-variant loci that meet certain requirements, or troubleshooting some unexpected results, to name a few.

Note: gunzip the files because SelectVariants uses unzipped files

VARIANTFILTRATION

Filter variant calls based on INFO and/or FORMAT annotations
This tool is designed for hard-filtering variant calls based on certain criteria. Records are hard-filtered by changing the value in the FILTER field to something other than PASS. Filtered records will be preserved in the output unless their removal is requested in the command line.

SelectV
:https://software.broadinstitute.org/gatk/documentation/tooldocs/4.0.4.0/org_broadinstitute_hellbender_tools_walkers_variantutils_SelectVariants.php
VariantF:https://software.broadinstitute.org/gatk/documentation/tooldocs/4.0.4.0/org_broadinstitute_hellbender_tools_walkers_filters_VariantFiltration.php
Extra:https://gatkforums.broadinstitute.org/gatk/discussion/2806/howto-apply-hard-filters-to-a-call-set)

[ Commands ] Checklist created on 05/13/2019 15:14.

- [ ] mkdir VCFs
for file in `ls|grep vcf`; do mv $file ./VCFs/;done (data clean-up)
gunzip -c GenotypeVCFs.vcf.gz > GenotypeVCFs.vcf
head -10 GenotypeVCFs.vcf (check file) Select SNP's
- [ ] nano SelectVariantsSNP.sh #!/usr/bin/bash JOBNAME=SelectVariantsSNP #PBS -N $JOBNAME
#PBS -q long
#PBS -l walltime=01:30:00
#PBS -l nodes=1:ppn=24
#PBS -k oe
module load gatk-4.0.4.0
cd /nlustre/users/junior/OmegaFastq/fastq/SAM/BAM/Sorted/Recali
gatk SelectVariants -R
/nlustre/users/fourie/H.sapiens/gatk_resource_bundle/2.8/hg19/ucsc.hg19.fasta -V
GenotypeVCFs.vcf -select-type SNP -O VariantSNPs.vcf
- [ ] chmod +755 SelectVariantsSNP.sh
qsub SelectVariantsSNP.sh
- [ ] Select Indels
- [ ] nano SelectVariantsINDEL.sh #!/usr/bin/bash JOBNAME=SelectVariantsINDEL #PBS -N
$JOBNAME
#PBS -q long
#PBS -l walltime=01:30:00
#PBS -l nodes=1:ppn=24
#PBS -k oe
module load gatk-4.0.4.0
cd /nlustre/users/junior/OmegaFastq/fastq/SAM/BAM/Sorted/Recali
gatk SelectVariants -R
/nlustre/users/fourie/H.sapiens/gatk_resource_bundle/2.8/hg19/ucsc.hg19.fasta -V
GenotypeVCFs.vcf -select-type INDEL -O VariantINDELS.vcf
- [ ] chmod +755 SelectVariantsINDEL.sh
qsub SelectVariantsINDEL.sh
- [ ] mkdir RawVariants
for file in `ls|grep Select`; do mv $file ./RawVariants/;done
for file in `ls|grep INDELS`; do mv $file ./RawVariants/;done
for file in `ls|grep SNP`; do mv $file ./RawVariants/;done
for file in `ls|grep Genotype`; do mv $file ./VCFs/;done (data clean-up)
- [ ] VariantFitration SNP's & INDELs
- [ ] nano FilterINDELS.sh #!/usr/bin/bash JOBNAME=FilterINDELS #PBS -N $JOBNAME
#PBS -q long
#PBS -l walltime=01:30:00
#PBS -l nodes=1:ppn=24
#PBS -k oe
module load gatk-4.0.4.0
cd /nlustre/users/junior/OmegaFastq/fastq/SAM/BAM/Sorted/Recali/RawVariants
gatk VariantFiltration -R
/nlustre/users/fourie/H.sapiens/gatk_resource_bundle/2.8/hg19/ucsc.hg19.fasta -V
VariantINDELS.vcf.gz -O VariantINDELHF.vcf.gz -filter "QD < 2.0 || FS > 200.0 ||
ReadPosRankSum < -20.0" --filter-name "HF_Indels"

☐ chmod +755 FilterINDELS.sh
qsub FilterINDELS.sh
☐ nano FilterSNP.sh #!/usr/bin/bash JOBNAME=FilterSNPs #PBS -N $JOBNAME
#PBS -q long
#PBS -l walltime=01:30:00
#PBS -l nodes=1:ppn=24
#PBS -k oe
module load gatk-4.0.4.0
cd /nlustre/users/junior/OmegaFastq/fastq/SAM/BAM/Sorted/Recali/RawVariants
gatk VariantFiltration -R
/nlustre/users/fourie/H.sapiens/gatk_resource_bundle/2.8/hg19/ucsc.hg19.fasta -V
VariantSNPs.vcf.gz -O VariantSNPSHF.vcf.gz -filter "QD < 2.0 || FS > 60.0 || MQ < 40.0 ||
MQRankSum < -12.5 || ReadPosRankSum < -8.0" --filter-name "HF_SNPs"
☐ chmod +755 FilterSNP.sh
qsub FilterSNP.sh
☐ Important note:
Index file is not present when I ran the scriptHad to delete the tbi files and run bgzip on them after bzip used tabix to create the right files.

*[ Selections_of_variants__which_are_hard_filtered... ]* File uploaded on 05/14/2019 13:17.

*[ Selections_of_Variants__which_are_hard_filtered... ]* File uploaded on 05/14/2019 13:17.

Comments for step SelectVariants & VariantFiltration

*No comments*

Samples of task Raw variants

*No items*

Task created on 05/13/2019 15:15.

## Variant Effector Predictor (VEP)

*No due date*

Variant Effect Predictor :The tool that gives you the possibility to analyse the variants in your samples.

Task tags: VEP

Protocol created on 05/13/2019 15:15.

*No protocol description*

Created by Junior MP on 05/20/2019 08:22.

**Step 1:** Variant Effect Predictor
**Uncompleted**

The Ensembl Variant Effect Predictor is a powerful toolset for the analysis, annotation, and prioritization of genomic variants in coding and non-coding regions. It provides access to an extensive collection of genomic annotation, with a variety of interfaces to suit different requirements, and simple options for configuring and extending analysis. It is open source, free to use, and supports full reproducibility of results. The Ensembl

Variant Effect Predictor can simplify and accelerate variant interpretation in a wide range of study designs.

Taken from: https://genomebiology.biomedcentral.com/articles/10.1186/s13059-016-0974-4
More info on :http://grch37.ensembl.org/Help/View?id=484

[ Commands for VEP ] Checklist created on 05/15/2019 14:03.

- [ ] zcat VariantSNPSHF.vcf.gz |cut -f 1,2,3,4,5,6,7 |less -S > VariantSNPSHFzcat VariantINDELHF.vcf.gz |cut -f 1,2,3,4,5,6,7 |less -S > VariantINDELSHF ---------- zcat VariantSNPSHF.vcf.gz |grep -v '#' |wc -l = 1583 zcat VariantSNPSHF.vcf.gz | less -S > testSNP
cat testSNP | vcf-subset -c BRB101_S20_L001 -r |grep -v '#' |wc -l ==> 1580compare differences in files with the option in vcf-tools vcftools --vcf testSNP.vcf --gzdiff VariantSNPSHF.vcf.gz --diff-site --out in1_v_in2

*[ Controle_no_differences_in_vcf_files.png ]* File uploaded on 05/15/2019 14:17.

*[ Variant_effect_predictor_analysis_of_INDELS_.png ]* File uploaded on 05/20/2019 07:57.

*[ VEP_analysis_of_INDELS.png ]* File uploaded on 05/20/2019 07:57.

*[ VEP_analysis_of_INDELS.png ]* File uploaded on 05/20/2019 07:57.

*[ VEP_analysis_of_INDELS.png ]* File uploaded on 05/20/2019 07:57.

*[ Variant_effect_predictor_analysis_of_SNP's.png ]* File uploaded on 05/20/2019 08:22.

*[ VEP_analysis_of_SNP's.png ]* File uploaded on 05/20/2019 08:22.

*[ VEP_analysis_of_SNP's.png ]* File uploaded on 05/20/2019 08:22.

Comments for step Variant Effect Predictor

*No comments*

Samples of task Variant Effector Predictor

(VEP)

*No items*

Task created on 05/20/2019 07:12.

## BCBIO

*No due date*

*No description*

Task tags: BCBIO

Protocol created on 05/20/2019 07:12.

Bcbio-nextgen provides best-practice pipelines for automated analysis of high throughput sequencing data with the goal of institutions/researchers using bcbio-nextgen for solving biological problems.

Created by Junior MP on 05/20/2019 09:53.

**Step 1:** Germ line variant calling (BCBIO)
**Uncompleted**

### Germ line variants calling:

Bcbio implements configurable SNP, indel and structural variant calling for germline populations. The software includes whole genome and exome evaluations against reference calls from the Genome in a Bottle consortium and Illumina Platinum Genomes project, enabling continuous assessment of new alignment and variant calling algorithms. The authors also regularly report on these comparisons and continue to improve approaches as the community makes new tools available.
The software automates post-variant calling annotation to make the outputs easier to feed directly into your biological analysis. It annotates variant effects using snpEff or Variant Effect Predictor (VEP), and prepare a GEMINI database that associates variants with multiple external annotations in a SQL-based query interface. GEMINI databases have the most associated external information for human samples (GRCh37/hg19 and hg38) but are available for any organism with the database populated using the VCF INFO column and predicted effects.

Taken from: https://bcbio-nextgen.readthedocs.io/en/latest/contents/pipelines.html
More info on: https://bcbio-nextgen.readthedocs.io/en/latest/index.html

[ Commands to run BCBIO ]

Checklist created on 05/20/2019 09:53.

- [ ] cd /nlustre/users/junior/
mkdir bcbio
mkdir fastq
cp /nlustre/users/junior/OmegaFastq/fastq/*.fastq
cd /nlustre/users/junior/bcbio/
mv cftr-gatk-template.yaml bc-gatk-template.yaml
cd /nlustre/users/junior/bcbio/
cd final/
cd 2019-05-17_bc
zcat bc-gatk-haplotype-joint-annotated.vcf.gz|grep -v "#" | grep -i "PASS"| cut -f 1,2,3,4,5,6,7 | less -S

Comments for step Germ line variant calling (BCBIO)

*No comments*

Created by Junior MP on 05/20/2019 12:21.

**Step 2:** MultiQC **Uncompleted**

### Germ line variants calling:

Bcbio implements configurable SNP, indel and structural variant calling for germline populations. The software includes whole genome and exome evaluations against reference calls from the Genome in a Bottle consortium and Illumina Platinum Genomes project, enabling continuous assessment of new alignment and variant calling algorithms. The authors also regularly report on these comparisons and continue to improve approaches as the community makes new tools available.
The software automates post-variant calling annotation to make the outputs easier to feed directly into your biological analysis. It annotates variant effects using snpEff or Variant Effect Predictor (VEP), and prepare a GEMINI database that associates variants with multiple external annotations in a SQL-based query interface. GEMINI databases have the most associated external information for human samples (GRCh37/hg19 and hg38) but are available for any organism with the database populated using the VCF INFO column and predicted effects.

Taken from: https://bcbio-nextgen.readthedocs.io/en/latest/contents/pipelines.html
More info on: https://bcbio-nextgen.readthedocs.io/en/latest/index.html

*[ MultiQC___General_statistics_all_samples.png ]* File uploaded on 05/20/2019 12:21.

*[ MultiQC_____mapped_to_reference_genome.png ]* File uploaded on 05/20/2019 12:21.

*[ MultiQC___chrXY_mapped_reads.png ]* File uploaded on 05/20/2019 12:21.

*[ MultiQC___Mapped_reads_per_config.png ]* File uploaded on 05/20/2019 12:21.

*[ MultiQC___Variant_effects_by_class_(Blue___sile... ]* File uploaded on 05/20/2019 12:21.

*[ MultiQC___FastQC_of_the_sequence_counts.png ]* File uploaded on 05/20/2019 12:21.

*[ MultiQC___FastQC__mean_quality_scores.png ]* File uploaded on 05/20/2019 12:21.

*[ MultiQC___FastQC__per_sequence_quality_scores.png ]* File uploaded on 05/20/2019 12:21.

Comments for step
MultiQC

*No comments*

Created by Junior MP on 05/20/2019
12:25.

---

**Step 3:** Analysis of all variants (VEP)
**Uncompleted**

*No description*

Comments for step Analysis of all variants
(VEP)

*No comments*

Samples of task
BCBIO

*No items*