**Hereditary Spastic Paraparesis**

Hereditary spastic paraparesis (HSP) is a neurodegenerative disorder estimated to affect 9.6 out of 100,000 individuals. Symptoms include severe intellectual disability, fluctuating central hypoventilation, gastresophageal reflux disease, wake apnea, areflexia, and unique dysmorphic features.

Exome sequencing of 5 individual patients coming from three Jewish Bukharian families identified a shared mutation in the gene TECPR2 (Oz-Levi D et. al. Am J Hum Genet. 2012 Dec 7;91(6):1065-72). The study is a joint effort of Tel-Hashomer hospital, the Weizmann Institute and Duke University.

**In the exercise, we will call genetic variants from two of the HSP patients, annotate the variants and identify the disease causing mutation (if time will allow).**

**The input files are located at course\_2018/HSP. To save computation time, the input fastq files contain only reads from chr 14.**

**Step 1: Make a folder to work and to store the results**

mkdir variants

cd variants

**Step 2: Load the required software into the environment**

We will work with bwa, picard, samtools, GATK and java (GATK and picard are written in java). Please load the following modules into the environment

module load bwa/0.7.15

module load samtools/1.3.1

module load picard/2.8.3

module load GATK/3.7

module load jdk/8.111

module load vcftools

**Step 3: Alignment using bwa**

We will use bwa to align the data to the human genome. The commands are stored in the files bwa\_command1.sh and bwa\_command2.sh (under course\_2018/HSP). Please submit the following 2 commands to wexac:

bsub -n 4 -q bio-guest -o sample1\_bwa.log.txt -e sample1\_bwa.err.txt –R \ "span[hosts=1]" ../course\_2018/HSP/bwa\_command1.sh

bsub -n 4 -q bio-guest -o sample2\_bwa.log.txt -e sample2\_bwa.err.txt –R \ "span[hosts=1]" ../course\_2018/HSP/bwa\_command2.sh

The commands you just submitted are composed from 3 steps: alignment (output is sam file), sam to bam conversion, and sorting. They look something like:

**bwa mem** -t 4 -R "@RG\tID:L001\tSM: HSP001" genome.fa R1.fastq R2.fastq **|** **samtools view** -Shu - **|samtools sort** -o HSP1.sort.bam

Where: the commands are **in bold**, followed by the parameters. The three commands are connected such that the output from one program will become the input of the next program. This was done using a pipe, by putting a vertical bar | between the commands. This saves a lot of I/O time, and disk space, as we don’t generate so many intermediate files.

Make sure that the alignment process is finished (using bjobs), and browse the log file to validate that it was successful.

**Step 4: Marking PCR duplicates**

We will use picard to mark PCR duplicates. The commands are stored in the files run\_picard\_1.sh and run\_picard\_2.sh. Please submit them to the server using:

bsub -q bio-guest -R "rusage[mem=4000]" -o picard1.log -e picard1.err.txt \ ../course\_2018/HSP/run\_picard\_1.sh

bsub -q bio-guest -R "rusage[mem=4000]" -o picard2.log -e picard2.err.txt \ ../course\_2018/HSP/run\_picard\_2.sh

Check the quality of the alignment using samtools flagstat

samtools flagstat sample1.picard.bam > sample1.alignment\_report.txt

samtools flagstat sample2.picard.bam > sample2.alignment\_report.txt

Have a brief look in the quality of the alignment. How many of the reads were aligned? If too many reads were not aligned it might indicate on a problem in the data. The number of marked duplicate reads is relatively high. This is a result of a relatively high coverage, and perhaps small insert size. Still, the quality of the data is high and allows variant calling.

more sample1.alignment\_report.txt

more sample2.alignment\_report.txt

**Step 5: Indexing and disk cleaning**

GATK requires indexed bam files as input. Index the alignment files using:

samtools index sample1.picard.bam

samtools index sample2.picard.bam

Also, remove the initial alignment, to save disk space

rm sample1.sort.bam

rm sample2.sort.bam

**Step 6: Variant calling with GATK to generate gvcf (without BSQR)**

bsub -q bio-guest -o GATK1.log -e GATK\_err1.txt -R "rusage[mem=4000]" \ ../course\_2018/HSP/GATK\_HC\_1.sh

bsub -q bio-guest -o GATK2.log -e GATK\_err2.txt -R "rusage[mem=4000]" \ ../course\_2018/HSP/GATK\_HC\_2.sh

The command looks something like (don’t try it!):

java -jar GenomeAnalysisTK.jar -T HaplotypeCaller -R genome.fasta -I yourbamfile -o output.g.vcf -L codingRegions.bed

Here we call the module HaplotypeCaller (using the –T HaplotypeCaller) to generate a gvcf file (which must ends with .g.vcf). Please note that to save time, we ask HaplotypeCaller to call variants only in specific regions that are defined in the file codingRegions.bed.

The output of the run is a gVCF file. Have a look in the gVCF files. Looks gibberish. The files store sequencing quality information for both variant and non-variant positions.

more sample1.g.vcf

more sample2.g.vcf

**Step 7: combine gVCF files**

bsub -q bio-guest -R "rusage[mem=8000]" -o joint\_calling.log –e \ joint\_calling.err.txt ../course\_2018/HSP/joint\_calling.sh

Note that this step is much faster than the previous. Browse the result:

more joined.vcf

**Step 8: variant filtration**

The vcf file still requires some filtration. We will use vcftools to remove low quality variants.

First, let’s count how many variants were initially obtained using vcftools:

vcftools --vcf joined.vcf

Apply filtering using

vcftools --vcf joined.vcf --minQ 40 --recode --out joined.filtered

Here we tell vcftools to remove all variants with quality < 40 using the parameter --minQ. The --recode tells the software to generate a new vcf file, and the –out joined.filtered defines the prename of the output file (joined.filtered.recode.vcf). As we work only with one chromosome, only one variant is filtered out.

**Download the bam files to your PC, open your local IGV to browse the predicted mutations. Have a look in at least 2 variations. Do they look real?**

**Step 9: variant annotation**

Copy the output file (joined.filtered.recode.vcf) to your PC and annotate the variants using Variant Effect Predictor of Ensembl.

How?

Open a browser with http://www.ensembl.org/info/docs/tools/vep/index.html and launch the web interface tool. The tool by default works with human version GRCh38.p12. Because we aligned the data to GRCh37,

go to GRCh37 website (press the linked text) and upload your vcf file.

Use the default setting except:

Transcript database to use: choose RefSeq transcripts

Filtering options: Restrict results: Show one selected consequence per variant

Run the process. View the results when the process is finished.

The top panel contains a summary with some basic statistics. Below you will see the results preview table. Each row is a variant (using the above setup, otherwise a row show a variant per transcript), with the annotation features of the variant. Have a look in the table. Can you identify the disease causing mutation? (hint: you can filter the result to see only variants with the symbol TECPR2). A disease-causing variant is expected to be rare in the population, and pathogenic. Is this the case?