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 the two HSP patients, annotate the variants and identify the disease causing mutation (if time will allow).

The input files are located at course_2017/HSP

Step 1: Make a folder to store the results

```
mkdir 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.txt and bwa_command2.txt (under course_2017/HSP). Please submit the commands to wexac:

```
bsub -n 4 -q bio-guest -o sample1_bwa.log.txt -e sample1_bwa.err.txt -R "span[hosts=1]"
course_2017/HSP/bwa_command1.txt
bsub -n 4 -q bio-guest -o sample2_bwa.log.txt -e sample2_bwa.err.txt -R "span[hosts=1]"
course_2017/HSP/bwa_command2.txt
```

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

a. Bwa alignment:

```
bwa mem -t 4 -R "@RG\tID: L001 \tSM: sample1" genome.fa R.fastq > out.sam
```

Here **genome.fa** is the human genome, **R.fastq** is a fastq file that contains the two paired-ends. In case you have two fastq files R.fastq can be replaced with **R1.fastq R2.fastq**
Note that we also told bwa to attach a read group identifier to each read (-R **"@RG\tID: L001 \tSM: sample1**"). The read group defines the lane of every read. GATK will not work without a definition of the read group.

b. sam to bam

```
samtools view -hb -o out.bam out.sam
```

c. Bam sort

```
samtools sort -o out.sort.bam out.bam
```

In bwa_command1.txt we connected all the steps together so 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.

e.g

```
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
```

Step 4: Removing of PCR duplicates

We will use picard to remove PCR duplicates. The commands are stored in the files run_picard_1.txt and run_picard_2.txt, please submit them to the server using:

```
bsub -q bio-guest -R "rusage[mem=4000]" -o picard1.log -e picard1.err.txt
course_2017/HSP/run_picard_1.txt
```

```
bsub -q bio-guest -R "rusage[mem=4000]" -o picard2.log -e picard2.err.txt
course_2017/HSP/run_picard_2.txt
```

Check the quality of the alignment using samtools flagstat

```
samtools flagstat variants/sample1.picard.bam > variants/sample1.alignment_report.txt
samtools flagstat variants/sample2.picard.bam > variants/sample2.alignment_report.txt
```

Have a brief look in the quality of the alignment.

```
more variants/sample1.alignment_report.txt
more variants/sample2.alignment_report.txt
```

Step 5: Indexing and disk cleaning

Index the alignment files, as is required by the GATK algorithm.

```
samtools index variants/sample1.picard.bam
samtools index variants/sample2.picard.bam
```

It is also advised to remove the initial alignment, to save disk space

```
rm variants/sample1.sort.bam
rm variants/sample2.sort.bam
```

Step 6: call variants using haplotype caller of GATK to generate gvcf (without BSQR)

```
bsub -q bio-guest -o GATK1.log -e GATK_err1.txt -R "rusage[mem=4000]"
course_2017/HSP/GATK_HC_1.txt
```

```
bsub –q bio-guest -o GATK2.log -e GATK_err2.txt -R "rusage[mem=4000]"
course_2017/HSP/GATK_HC_2.txt
```

```
The command is:
```

```
java -Xmx4g -jar GenomeAnalysisTK.jar -T HaplotypeCaller -R genome.fasta -I yourbamfile -
o output.g.vcf -L codingRegions.bed -ERC GVCF
```

Here we call the module HaplotypeCaller (using the –T) 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

```
more variants/sample1.alignment_report.txt
more variants/sample1.alignment_report.txt
```

Step 7: combine gVCF files

```
bsub -q bio-guest -R "rusage[mem=16000]" -o joint_calling.log -e joint_calling.err.txt
course_2017/HSP/joint_calling.txt
```

Browse the result:

```
more variants/joined.vcf
```

Step 8: variant filtration

We can use vcftools to remove low quality variants.

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

`vcftools –vcf variants/joined.vcf`

Apply filtering using

```
vcftools --vcf joined.vcf --minQ 20 --recode --out joined.filtered
```

Here we tell vcftools to remove all variants with quality < 20. 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).

Open IGV to browse the predicted mutations. Have a look in at least 2.

Step 9: variant annotation

Copy the output file to your PC and annotate the variants using Variant Effect Predictor of Ensembl.

Open a browser with `http://www.ensembl.org/info/docs/tools/vep/index.html` and launch the tool.

Go to GRCh37 website => Upload your vcf file => and press the run button.

Can you identify the disease causing mutation? (hint: you can filter the result to see only variants with the symbol TECPR2)