Introduction to Variant calling analysis

Detection of small variants (i.e., SNPs, Indels, Deletions) in Whole-Exome/-Genome Sequencing data

Exome & Whole-genome Sequencing

- Whole-exome Sequencing (WES) is a method that enables the selective sequencing of the exonic regions of a genome (i.e., mRNAs & UTRs)
- ~180.000 exons in human genome, representing only ~1% but harboring up to 85% of all disease-causing variants (Choi et al., 2009)
- Whole-genome Sequencing (WGS) encompasses the entire length of the genome
- WES vs WGS:
  - WGS provides more data but requires additional time to process
  - WES captures most information in a cost-effective way
  - WGS more suitable for Copy-number variation (CNV) analysis
- Notably, the costs of WES may actually not be higher even today than the costs of conventional genetic testing (Vissers et al., 2017)

In this tutorial

- Raw WES data (fastq) from a family trio
- The boy child (sample name: “proband”) is affected by the disease osteopetrosis
- Both parents, who happen to be consanguineous, are unaffected
- We will go through the steps of a typical variant calling analysis
- Our goal is to identify the genetic variation that is responsible for the disease
- (Optional) Explore the original workflow using Galaxy tutorial (Galaxy Europe tools only)

FastQ Format (doc)

- FASTQ format stores sequences and Phred qualities in a single file.
- It is concise and compact.
- FASTQ is first widely used in the Sanger Institute and therefore we usually take the Sanger specification and the standard FASTQ format, or simply FASTQ format.
- Although Solexa/Illumina read file looks pretty much like FASTQ, they are different in that the qualities are scaled differently. In the quality string, if you can see a character with its ASCII code higher than 90, probably your file is in the Solexa/Illumina format.

FastQ Format (doc)

- Format:
  - @<seqname>
  - <sequence>
  - +
  - <quality>
- Requirements
  - The <seqname> following '+' is optional, but if it appears right after '+', it should be identical to the <seqname> following '@'.
  - The length of <sequence> is identical the length of <quality>. Each character in <quality> represents the phred quality of the corresponding nucleotide in <sequence>.
  - The <quality> field represent the Phred quality score (non-negative integer) for each nucleotide, encoded here as a character based on the ASCII table.

Example

```
@EAS54_6_R1_2_1_413_324
CCCTTCTTGTCTTCAGCGTTTCTCC
+
;;3;;;;;;;;;;;;7;;;;;;;88
@EAS54_6_R1_2_1_540_792
TTGGCAGGCCAAGGCCGATGGATCA
+
;;;;;;;;;;;7;;;;;-;;;3;83
@EAS54_6_R1_2_1_443_348
GTTGCTTCTGGCGTGGGTGGGGGGG
+EAS54_6_R1_2_1_443_348
;;;;;;;;;;;9;7;;.7;393333
```

Steps prior to Variant detection

- Quality control
- Trimming of low-quality and/or adapter sequences
- Mapping reads to reference genome
- Post-processing of mapped reads
  - SAM-to-BAM format conversion
  - Generating mapping summary statistics
  - Filtering reads based on SAM/BAM flags
  - Add information in BAM files
  - Mark duplicated reads
  - Generate reference indexes

Steps prior to Variant detection

- Quality control
- Trimming of low-quality and/or adapter sequences
- Mapping reads to reference genome
- Post-processing of mapped reads
  - SAM-to-BAM format conversion
  - Generating mapping summary statistics
  - Filtering reads based on SAM/BAM flags
  - Add information in BAM files
  - Mark duplicated reads
  - Generate reference indexes

Steps prior to Variant detection

- Quality control
- Trimming of low-quality and/or adapter sequences
- **Mapping reads to reference genome**
- Post-processing of mapped reads
  - SAM-to-BAM format conversion
  - Generating mapping summary statistics
  - Filtering reads based on SAM/BAM flags
  - Add information in BAM files
  - Mark duplicated reads
  - Generate reference indexes

SAM & BAM format

```
@HD VN:1.6 SO:coordinate
@SQ SN:ref LN:45
r001   99 ref  7 30 8M2I4M1D3M = 37  39 TTAGATAAAGGATACTG *
r002    0 ref  9 30 3S6M1P1I4M *  0   0 AAAAGATAAGGATA    *
r003    0 ref  9 30 5S6M       *  0   0 GCCTAAGCTAA       * SA:Z:ref,29,-,6H5M,17,0;
r004    0 ref 16 30 6M14N5M    *  0   0 ATAGCTTCAGC       *
r003 2064 ref 29 17 6H5M       *  0   0 TAGGC             * SA:Z:ref,9,+,5S6M,30,1;
r001  147 ref 37 30 9M         =  7 -39 CAGCGGCAT         * NM:i:1
```

- SAM stands for Sequence Alignment/Map format.
- It is a TAB-delimited text format consisting of a header section, which is optional, and an alignment section. If present, the header must be prior to the alignments.
- Header lines start with '@', while alignment lines do not. Each alignment line has 11 mandatory fields for essential alignment information such as mapping position, and variable number of optional fields for flexible or aligner specific information.
- A BAM file (*.bam) is the compressed binary version of a SAM file that is used to represent aligned sequences up to 128 Mb.
- SAM and BAM formats are described in detail at https://samtools.github.io/hts-specs/SAMv1.pdf.

Steps prior to Variant detection

- Quality control
- Trimming of low-quality and/or adapter sequences
- Mapping reads to reference genome
- Post-processing of mapped reads
  - SAM-to-BAM format conversion
  - Generating mapping summary statistics
  - Filtering reads based on SAM/BAM flags
  - Add information in BAM files
  - Mark duplicated reads
  - Generate reference indexes

Steps prior to Variant detection

- Quality control
- Trimming of low-quality and/or adapter sequences
- Mapping reads to reference genome
- Post-processing of mapped reads
  - SAM-to-BAM format conversion
  - Generating mapping summary statistics
  - Filtering reads based on SAM/BAM flags
  - Add information in BAM files
  - Mark duplicated reads
  - Generate reference indexes

Steps prior to Variant detection

- Quality control
- Trimming of low-quality and/or adapter sequences
- Mapping reads to reference genome
- Post-processing of mapped reads
  - SAM-to-BAM format conversion
  - Generating mapping summary statistics
  - Filtering reads based on SAM/BAM flags
  - Add information in BAM files
  - Mark duplicated reads
  - Generate reference indexes

SAM & BAM format – the alignment section

Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	`[!-?A-~]{1,254}`	Query template NAME
2	FLAG	Int	$[0, 2^{16} - 1]$	bitwise FLAG
3	RNAME	String	`\*|[:rname:^*=][:rname:]*`	Reference sequence NAME[11]
4	POS	Int	$[0, 2^{31} - 1]$	1-based leftmost mapping POSition
5	MAPQ	Int	$[0, 2^{8} - 1]$	MAPping Quality
6	CIGAR	String	`\*|([0-9]+[MIDNSHPX=])+`	CIGAR string
7	RNEXT	String	`\*|=|[:rname:^*=][:rname:]*`	Reference name of the mate/next read
8	PNEXT	Int	$[0, 2^{31} - 1]$	Position of the mate/next read
9	TLEN	Int	$[-2^{31} + 1, 2^{31} - 1]$	observed Template LENgth
10	SEQ	String	`\*|[A-Za-z=.]+`	segment SEQuence
11	QUAL	String	`[!-~]+`	ASCII of Phred-scaled base QUALity+33

- https://broadinstitute.github.io/picard/explain-flags.html

Steps prior to Variant detection

- Quality control
- Trimming of low-quality and/or adapter sequences
- Mapping reads to reference genome
- Post-processing of mapped reads
  - SAM-to-BAM format conversion
  - Generating mapping summary statistics
  - Filtering reads based on SAM/BAM flags
  - Add sample information in BAM files
  - Mark duplicated reads
  - Generate reference indexes

Steps prior to Variant detection

- Quality control
- Trimming of low-quality and/or adapter sequences
- Mapping reads to reference genome
- Post-processing of mapped reads
  - SAM-to-BAM format conversion
  - Generating mapping summary statistics
  - Filtering reads based on SAM/BAM flags
  - Add information in BAM files
  - Mark duplicated reads
  - Generate reference indexes

Steps prior to Variant detection

- Quality control
- Trimming of low-quality and/or adapter sequences
- Mapping reads to reference genome
- Post-processing of mapped reads
  - SAM-to-BAM format conversion
  - Generating mapping summary statistics
  - Filtering reads based on SAM/BAM flags
  - Add information in BAM files
  - Mark duplicated reads
  - Generate indexes

Variant detection

- We will use ***FreeBayes*** to call our variants
- *FreeBayes* is a Bayesian genetic variant detector designed to find small polymorphisms, specifically SNPs (single-nucleotide polymorphisms), indels (insertions and deletions), MNPs (multi-nucleotide polymorphisms), and complex events (composite insertion and substitution events) smaller than the length of a short-read sequencing alignment
- Similar software tools include:
  - GATK HaplotypeCaller
  - GATK Mutect2 (emphasis on somatic variants)
  - bcftools
  - varscan2

VCF format

- VCF is a text file format (most likely stored in a compressed manner).
- It contains meta-information lines, a header line, and then data lines each containing information about a position in the genome.
- The format also has the ability to contain genotype information on samples for each position.
- 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 detailed information about the VCF format are available here: https://samtools.github.io/hts-specs/VCFv4.2.pdf

VCF format

```
##fileformat=VCFv4.2
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=file:///seq/references/1000GenomesPilot-NCBI36.fasta
##contig=<ID=20,length=62435964,assembly=B36,md5=f126cdf8a6e0c7f379d618ff66beb2da,species="Homo sapiens",taxonomy=x>
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, build 129">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM POS     ID        REF    ALT     QUAL FILTER INFO                              FORMAT      NA00001        NA00002        NA00003
20    14370   rs6054257 G      A       29   PASS   NS=3;DP=14;AF=0.5;DB;H2           GT:GQ:DP:HQ 0|0:48:1:51,51 1|0:48:8:51,51 1/1:43:5:.,.
20    17330   .         T      A       3    q10    NS=3;DP=11;AF=0.017               GT:GQ:DP:HQ 0|0:49:3:58,50 0|1:3:5:65,3   0/0:41:3
20    1110696 rs6040355 A      G,T     67   PASS   NS=2;DP=10;AF=0.333,0.667;AA=T;DB GT:GQ:DP:HQ 1|2:21:6:23,27 2|1:2:0:18,2   2/2:35:4
20    1230237 .         T      .       47   PASS   NS=3;DP=13;AA=T                   GT:GQ:DP:HQ 0|0:54:7:56,60 0|0:48:4:51,51 0/0:61:2
20    1234567 microsat1 GTC    G,GTCT  50   PASS   NS=3;DP=9;AA=G                    GT:GQ:DP    0/1:35:4       0/2:17:2       1/1:40:3
```

Steps following Variant detection

- Compress and index VCF files
- Apply filters to the detected variants
- Merge the VCF files of all samples
- Format merged VCF files for further analysis
- Annotation of the detected variants

Steps following Variant detection

- Compress and index VCF files
- Apply filters to the detected variants
- Merge the VCF files of all samples
- Format merged VCF files for further analysis
- Annotation of the detected variants

Steps following Variant detection

- Compress and index VCF files
- Apply filters to the detected variants
- Merge the VCF files of all samples
- Format merged VCF files for further analysis
- Annotation of the detected variants

Steps following Variant detection

- Compress and index VCF files
- Apply filters to the detected variants
- Merge the VCF files of all samples
- Format merged VCF files for further analysis
- Annotation of the detected variants

Steps following Variant detection

- Compress and index VCF files
- Apply filters to the detected variants
- Merge the VCF files of all samples
- Format merged VCF files for further analysis
- Annotation of the detected variants

Validate results - GEMINI analysis

- Access GEMINI through Galaxy Europe
- SQL database with annotated VCF file using GEMINI
- Candidate variant detection by testing for the inheritance pattern “Autosomal recessive”
- A table of candidate variants along with their respective p-values.
- Mutation responsible for the phenotype in this analysis:

max_aaf_all	chrom	start	ref	alt	impact	gene	clinvar_sig	clinvar_disease_name	clinvar_gene_phenotype	rs_ids	variant_id	family_id	family_members	family_genotypes	samples	family_count
3,2489E-05	chr8	86385979	G	A	stop_gained	CA2	None	None	carbonic_anhydrase_ii_variant|osteopetrosis_with_renal_tubular_acidosis	None	3297	FAM	father(father;unaffected;male),mother(mother;unaffected;female),proband(proband;affected;male)	G/A,G/A,A/A	proband	1

Validate results - GEMINI load

1. **GEMINI load** 🔧 with
   - 🗋 *"VCF dataset to be loaded in the GEMINI database"*: the output of **SnpEff eff** 🔧
   - *"The variants in this input are"*: `annotated with snpEff`
   - *"This input comes with genotype calls for its samples"*: `Yes`

     Sample genotypes were called by Freebayes for us.
   - *"Choose a gemini annotation source"*: select the latest available annotations snapshot (most likely, there will be only one)
   - *"Sample and family information in PED format"*: the pedigree file prepared above
   - *"Load the following optional content into the database"*
     - ☑ *"GERP scores"*
     - ☑ *"CADD scores"*
     - ☑ *"Gene tables"*
     - ☑ *"Sample genotypes"*
     - ☑ *"variant INFO field"*

     Leave **unchecked** the following:

     - *"Genotype likelihoods (sample PLs)"*

       Freebayes does not generate these values
     - *"only variants that passed all filters"*

       This setting is irrelevant for our input because Freebayes did not apply any variant filters.

Validate results - GEMINI inheritance pattern

1. **GEMINI inheritance pattern** 🔧
   - *"GEMINI database"*: the GEMINI database of annotated variants; output of **GEMINI load** 🔧
   - *"Your assumption about the inheritance pattern of the phenotype of interest"*: `Autosomal recessive`
     - ⊞ *"Additional constraints on variants"*
       - *"Additional constraints expressed in SQL syntax"*: `impact_severity != 'LOW'`

         This is a simple way to prioritize variants based on their functional genomic impact. Variants with *low impact severity* would be those with no obvious impact on protein function (*i.e.*, silent mutations and variants outside coding regions)
     - *"Include hits with less convincing inheritance patterns"*: `No`

       This option is only meaningful with larger family trees to account for errors in phenotype assessment.
     - *"Report candidates shared by unaffected samples"*: `No`

       This option is only meaningful with larger family trees to account for alleles with partial phenotypic penetrance.
   - *"Family-wise criteria for variant selection"*: keep default settings

     This section is not useful when you have data from just one family.
   - In *"Output - included information"*
     - *"Set of columns to include in the variant report table"*: `Custom (report user-specified columns)`
       - *"Choose columns to include in the report"*:
         - ☑ *"alternative allele frequency (max_aaf_all)"*
       - *"Additional columns (comma-separated)"*: `chrom, start, ref, alt, impact, gene, clinvar_sig, clinvar_disease_name, clinvar_gene_phenotype, rs_ids`