Practicum Bioinformatics - answer key

Prof. Kris Laukens - Adrem Data Lab
Department of Computer Science, University of Antwerp

Academic year 2024-2025

Contents

Note about the answers

These are just example solutions. For many of the exercises there will be other correct solutions as well. The main goal of these practicals is to become acquainted with the different databases, techniques and principles of bioinformatics. Try to understand why we are using a specific technique, the different aspects of the outputs of the tools and how they work, rather than learning everything by heart. You will need to be able to interpret related questions to the ones you've solved here while using this text as a quick reference, but you will need to know the major sections and themes of the practicals if you want to be able to retrieve them in a timely fashion.

1 Next Generation Sequencing: Command line tutorial

Section 9: The FASTQ file format

We will start this practicum with an exercise on the FastQ format. A FastQ file is a specific file that is generated by the sequencer and provided to the user. It is actually a text file, but has a specific structure as described below and we have seen during the course. It is the description of the millions of small fragments the sequencer has sequenced. The data generated for 1 DNA fragment or 1 `"read"` corresponds to 4 lines in the FASTQ, structured as follows:

- Line 1: `'@ + description'`. The description contains information about the machine, location, etc.
- Line 2: the sequence
- Line 3: `'+'` and can be followed by line 1 again
- Line 4: Phred scores converted to ASCII codes
  - **Problem!** Different companies use different ASCII encoding. Illumina by itself already uses three different types. Choosing the wrong off-set could completely ruin your analysis!

The first line contains the description of the read. This includes the name of the machine, the location of the read in the flow cell, etc. Line 2 contains the sequence that the sequencer detected for the fragment (A/C/T/G or N when the sequencer could not determine the base). Line 3 is utterly useless and contains a `"+"` and optionally a repeat of the information of line 1. Line 4 contains the Phred Scores converted to ASCII code. This line is of equal length as Line 2 and each ASCII symbol in line 4 corresponds to exactly 1 base in line2. Below you can find a table of ASCII characters, only look at the first (Dec) and the last (Chr, in red) column.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

As you have seen during the course, different ASCII offsets can be used during NGS.

ASCII offsets used in NGS:

- Sanger and newest Illumina machines (>1.8): Offset +33
- Solexa/Illumina 1.0: +59
- Illumina 1.3 -1.8: +64

So for example, in Sanger coding (which is most often used) ASCII symbol 33, which is "!" corresponds to quality score 0, since the offset is 33. "?" corresponds to ASCII symbol 63 and therefore to quality 63-33 = 30. In case of Illumina coding, the offset is 64, and the "@" symbol corresponds to Phred score 0. Since you cannot have negative Phred Scores, and the Phred scores are maximum 40, some symbols cannot be encountered in certain codings. As you can see, the different codings can be quite confusing. Therefore new sequencing data is always in Illumina format. It is however important to remember the different coding systems, since you can always come across an older dataset when performing research.

You can easily calculate Phred scores back to probabilities:

$$Q = -10 \log_{10} P$$

where Q is the Q-score and P is the probability the base is wrong.

Determine the phred quality scores of the underlined bases, given these are Illumina reads (offset = +33). What is the probability that these bases are wrong?

```
@M00984:14:000000000-AA0HF:1:1101:23031:1298 1:N:0:1
CGTGCCAACGGCACTCGTACACGAGTTGTACAGAAACTGAT
+
CCCCCGGGGGGGGGGGGGGGGGFDGGGGGGGFFGGGGG9F
```

- The highlighted and underlined G corresponds to the ASCII-encoded quality score "C". "C" corresponds to the value 67 in the ASCII table. The quality score Q is therefore: $67 - 33 = 34$
- The probability $= 10^{\frac{34}{-10}} = 0.000398$ (so there is a chance of 1 in 2512 that the base was called wrong).

- The A corresponds to the ASCII-encoded quality score "9". "9" is 57 in the ASCII table. $57 - 33 = 24$
- Therefore, 24 is the Q-score and the probability is $P = 10^{\frac{24}{-10}} = 0.00398$ (so a chance of 1 in 251 that the base is wrong).

What ASCII Offset was used below? Important information: The Q score never exceeds 40 or falls below 0.

```
@SRR038845.3 HWI-EAS038:6:1:0:1938 length=36
CAACGAGTTCACACCTTGGCCGACAGGCCCGGGTAA
+SRR038845.3 HWI-EAS038:6:1:0:1938 length=36
BA@7>B=>:>>7@7@>>9=BAA?;>52;>:9=8.=A
@SRR038845.41 HWI-EAS038:6:1:0:1474 length=36
CCAATGATTTTTTTCCGTGTTTCAGAATACGGTTAA
+SRR038845.41 HWI-EAS038:6:1:0:1474 length=36
BCCBA@BB@BBBBAB@B9B@=BABA@A:@693:@B=
@SRR038845.53 HWI-EAS038:6:1:1:360 length=36
GTTCAAAAAGAACTAAATTGTGTCAATAGAAAACTC
+SRR038845.53 HWI-EAS038:6:1:1:360 length=36
BBCBBBBBB@@BAB?BBBBCBC>BBBAA8>BBBAA@
```

There are digits (0-9) in the ASCII code (=ASCII 57 and below), therefore, it can not be offset 59 or 64 (since 59 and 64 are 0, and it can't go negative), and has to be 33, so Sanger encoding.

What ASCII Offset was used below (and so, which format is it)?

```
@HWI-EAS209_0006_FC706VJ:5:58:5894:21141#ATCACG/1
TTAATTGGTAAATAAATCTCCTAATAGCTTAGATNTTACCTTNNNNNNNNNNTAGTTTCTTG
+HWI-EAS209_0006_FC706VJ:5:58:5894:21141#ATCACG/1
efcfffffcfeefffcffffffddf`feed]`]_Ba_^__[YBBBBBBBBBBRTT\]][]dd
```

> Firstly, there is an indication that it could be Illumina 1.3 -1.8 (offset $+64$) since there are no ASCII symbols corresponding to a Dec value < 64 present. This is confirmed by the fact we have the symbol `"f"` for example. ASCII number for `"f"` is 102. Given that the maximum score of sequencing quality can be 40 with current sequencers: 102 - 40 $= 62 =>$ it can't be offset 33 or 59 because the score for symbol `"f"` would be over 40.

Section 10: Preparation

10.1 Introduction

In this tutorial, based on the Galaxy tutorial, we will guide you through the process of analyzing Next-Generation Sequencing (NGS) data for SARS-CoV-2 using command line tools, starting from FASTQ data. Understanding command line analysis, choosing the right tools and knowing how to use them, is crucial in bioinformatics. We will cover the following analysis steps, which will be further explained later on:

1. Downloading data
2. Quality control and trimming
3. Alignment
4. Duplicate removal
5. Realigning reads
6. Indel qualities
7. Indexing alignment
8. Variant calling
9. Annotating variant effects
10. Creating a table of variants
11. Summarizing data with MultiQC

10.2 Creating a conda environment

Conda is an open-source package management system and environment management system that simplifies the installation and management of software packages and their dependencies. In bioinformatics, where we often work with a variety of tools and packages, conda environments provide a convenient way to create isolated environments for different projects. Each conda environment is a self-contained directory that encapsulates a specific set of packages and their dependencies, allowing you to have multiple versions of the same package installed on your system without conflicts.

Conda environments offer several benefits for bioinformatics workflows. They promote reproducibility by enabling the creation of environments with specific package versions, ensure proper dependency management, provide isolation between projects to avoid package conflicts, facilitate portability and sharing of analysis pipelines, and integrate with Bioconda, a popular channel for bioinformatics packages. By leveraging conda environments, bioinformaticians can streamline their workflows, ensure consistency, and focus on the analysis rather than package management.

So before diving into the analysis, we need to set up a conda environment with the necessary tools. The following command creates a new environment named "ngs" and installs the required packages from the Bioconda channel:

```
conda create -n ngs -c bioconda sra-tools fastp bwa-mem2 \
samtools picard lofreq snpeff snpsift multiqc --yes
```

This command installs the tools for downloading data (sra-tools), quality control and trimming (fastp), alignment (bwa-mem2), BAM file manipulation (samtools), duplicate removal (picard), variant calling and realignment (lofreq), variant annotation (snpeff and snpsift), and generating a summary report (multiqc). The `-yes` flag automatically answers "yes" to any prompts during the installation process.

Make sure you activate the environment with

```
conda activate ngs
```

10.3 About sequencing data

To perform any analysis, we also need data. For this tutorial, we'll use one of the two datasets used in the Galaxy tutorial: `https://www.ebi.ac.uk/ena/browser/view/SRR12733957`, accession number SRR12733957. The data is Illumina derived from patients infected with SARS-CoV-2. In general, it is recommended to use sequencing reads from one exome. **Why not sequence the complete human genome, but look at exomes only?**

> The human genome is gigantic (gigabases or 10^9 bases), and the coding part is only a few megabases (10^6 bases). Therefore you can sequence many human exomes on one sequencer, but only one (or few) human genome. It is thus much more price and time efficient to sequence only exomes, if this is the only part you want to look at. For example, for studying genetic diseases this is often sufficient.

Single-end or paired-end data? Single-read sequencing involves the sequencing of DNA from just one end, which is the most straightforward method for sequencing. In contrast to single-read sequencing, paired-end sequencing enables the sequencing of both ends of a fragment, resulting in the production of higher-quality, alignable sequence data. This approach supports the identification of genomic rearrangements, repetitive sequence elements, as well as gene fusions and novel transcripts.

Beyond the advantage of generating twice the number of reads within the same time and effort in library preparation, sequences aligned as read pairs offer improved accuracy in read alignment. Additionally, they enhance the capacity to detect insertion-deletion (indel) variants, a task that proves more challenging with single-read data.

10.4 Downloading data

There are different methods for downloading data, such as using `wget`, a web browser, or specifically for SRA data, you can also use `fastq-dump`.

You could use fastq-dump in the following, simple way to download data:

```
fastq-dump [options] <accession number>
```

Depending whether the data you're downloading is single-end or paired-end, you'll need to add the `-split-files` option. Check `fastq-dump -help` for all the options.

If that doesn't work, you can also use `wget` to download data. Then it's just:

```
wget <url-to-data>
```

To make sure we're all on the same page regarding the data, you get the url to download the data here:

- `ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR127/057/SRR12733957/SRR12733957_1.fastq.gz`
- `ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR127/057/SRR12733957/SRR12733957_2.fastq.gz`

We will also need a reference genome of SARS-CoV-19 (NC_045512.2) to perform mapping with later on. We'll also use `wget`:

```
wget https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/009/858/895/GCF_009858895.2_ASM985889v3/GCF_009858895.2_ASM985889v3_genomic.fna.gz
```

The reference genome is in FASTA format, which is a standard format for representing nucleotide or amino acid sequences.

Section 11: NGS data logistics via command line

Now that we've got the data, we can start analyzing it!

11.1 Quality Control and Trimming

Assessing the quality of your sequencing data is the first step of NGS analysis: Do the reads contain low-quality portions? Is the sequenced library diverse enough (e.g frequent kmers present)? A great tool for assessing the quality of your data is FASTQC, another tool is fastp (which also allows for data preprocessing like trimming and filtering).

Trimming adapters and low-quality bases helps improve the alignment and variant calling steps. Preprocessing involves cutting away low quality (parts of) reads or adapter sequences. A strategy that is often used, to maintain only the high-quality parts of sequencing reads, is read trimming. When the quality scores start to drop under a user-defined Q-score threshold, the read is eventually chopped off, based on a slide window score function (we will not go into details here). Removing sequencing adapters improves alignments and variant calling. Common used tools here are **fastp**, **Trimmomatic** and **CutAdapt**.

For MultiQC or FASTQC, a report can be be generated before and after preprocessing to inspect if there is improvement in the read quality. Fastp already generates a before/after filtering comparison. We will use the `fastp` tool for quality control and trimming:

Configure the command by replacing '<input_R1>', '<input_R2>', '<output_R1>', and '<output_R2>' with the filenames for your input and output FASTQ files, respectively:

```
fastp -i <input_R1> -I <input_R2> -o <output_R1> -O <output_R2>
```

The input files (`<input_R1>` and `<input_R2>`) are the paired-end FASTQ files, and the output files (`<output_R1>` and `<output_R2>`) are the trimmed FASTQ files.

Questions:

1. Which other files get created in addition to the fastq files?
2. What is the percentage of reads that are retained?
3. What is the most frequent 5-mer (sequence of 5 nucleotides) in the R2 dataset before and after fastp quality control? Which 5-mers do you think are biologically relevant?

```
fastp -i in.R1.fq.gz -I in.R2.fq.gz -o out.R1.fq.gz -O out.R2.fq.gz
```

Question answers

1. A fastp report is generated, both in json and html format.
2. 67.98%, can most easily be found in the html report (figure).
3. Before: GGGGG, after AAAAA and TTTTT, the latter can be part of the polyA tail and thus biologically relevant, while polyG is most likely a sequencing artefact.

Filtering result	
reads passed filters:	594.676000 K (67.987525%)
reads with low quality:	279.876000 K (31.997384%)
reads with too many N:	132 (0.015091%)
reads too short:	0 (0.000000%)

11.2 Alignment (BWA-MEM2)

After checking the quality the data and performing preprocessing, the next step is mapping your NGS reads to reference sequences.

This step takes one by one the sequences from the FastQ files and checks on which position of the reference genome they match. Mappers usually compare reads against a reference sequence that has been transformed into a highly accessible data structure called genome index. Such indexes should be generated before mapping begins.

Mapping is one of the key steps of the analysis. Common used tools are **Bowtie2** and **BWA**.

Mapping results in a BAM file.

The BAM file, resulting from the mapping, contains all the information about the read that was also present in the FastQ file, but additionally it also contains information about the position where it matches to the reference genome, the pairwise alignment with this position, and the mapping score.

Explore some lines of the BAM file displayed in SAM format and make sure that you understand the file format. The Appendix: Description of file types, at the end of this document can help you.

We will use the BWA-MEM2 aligner, which is an improved version of the widely used BWA-MEM aligner. Follow these steps for alignment:

1. **Unzip the reference genome:** Unzip the reference genome file which is in gzip format. Replace `<reference_genome>.gz` with the actual filename of your compressed reference genome.

   ```
   gunzip <reference_genome>.gz
   ```

2. **Index the Reference Genome**:

   Index the reference genome using BWA-MEM2. Replace `<reference_genome>` with the filename of your uncompressed reference genome.

   ```
   bwa-mem2 index <reference_genome>
   ```

   ```
   bwa-mem2 index GCF_009858895.2_ASM985889v3_genomic.fna
   ```

3. **Perform alignment:** Align your trimmed sequencing reads to the reference genome. Replace `<reference_genome>`, `<forward_reads>.gz`, and `<reverse_reads>.gz` with the filenames of your reference genome and forward/reverse sequencing reads respectively.

   ```
   bwa-mem2 mem <reference_genome> <forward_reads>.gz <reverse_reads>.gz
   > <output>.sam
   ```

   ```
   bwa-mem2 mem GCF_009858895.2_ASM985889v3_genomic.fna \
   SRR12733957_1_trim.fq.gz SRR12733957_2_trim.fq.gz \
   > aligned_reads.sam
   ```

4. **Convert SAM to BAM Format**:

   Convert the alignment output from SAM format to BAM format for efficient storage and processing. Replace `<sam_file>` and `<bam_file>` with your input and output file names respectively.

   ```
   samtools view -bS <sam_file> -o <bam_file>
   ```

   Questions:

   - How does the filesize differ between the `.sam` and the `.bam` file?
   - What do the first 5 rows of the `.sam` file look like? Could you do the same for the `..bam`?

```
samtools view -bS aligned_reads.sam -o aligned_reads.bam
```

Answers

- 1. Using `ll -h`: 157 MB for the sam, to 34MB for the bam.
- 2. The sam is a fastq format, while the bam cannot be opened this way, it is a binary file.

5. **Sort the BAM File**: Sort the BAM file to prepare it for further analysis. Replace `<input_bam>` and `<sorted_bam>` with your input and desired sorted BAM file names respectively.

   ```
   samtools sort <input_bam> -o <sorted_bam>
   ```

   ```
   samtools sort -o sorted_aligns.bam aligned_reads.bam
   ```

11.3 Duplicate Removal

PCR steps during the library preparation can result in the same DNA-fragment being present several times in your library (PCR amplification). This can result in the same fragment being sequenced several times and this is a problem, since they are not true sequencing replicates: they originate from the same molecule. This problem becomes relatively more important with low concentrations of starting DNA and could potentially cause problems for the downstream analysis. For example, you could overestimate your genome coverage in certain regions (you assume you sequenced/sampled more DNA fragments), which could result in significantly different allele frequencies, while there are not.

A common used tool is MarkDuplicates from the Picard package. This tool will remove all reads with an identical start and stop position, except the one with the best mapping quality. This will solve the problem of PCR duplicates, however, there are also identical fragments in your library that originate from different DNA molecules (and are therefore true replicates) and these will also be removed.

Replace the placeholders with the actual filenames and ensure that the `REMOVE_DUPLICATES` option is set correctly based on the desired outcome:

```
picard MarkDuplicates \
INPUT=<input_bam> \
OUTPUT=<output_bam> \
METRICS_FILE=<metrics_file>.txt \
REMOVE_DUPLICATES=false
```

The `MarkDuplicates` tool identifies duplicate reads and marks them in the output BAM file. The `REMOVE_DUPLICATES` option is set to `false` to retain the duplicate reads with appropriate flags.

```
picard MarkDuplicates \
INPUT=sorted_aligns.bam \
OUTPUT=deduped_align.bam \
METRICS_FILE=markdup_metrics.txt \
REMOVE_DUPLICATES=false
```

11.4 Realigning Reads

Realignment around indels helps improve the accuracy of variant calling. We will use the `lofreq viterbi` tool for realigning reads:

Replace the filenames appropriately in the command below to perform the realignment:

`lofreq viterbi -f <reference_genome> -o <output_bam> <input_bam>`

The realigned reads are written to the specified `output_bam` file.

```
lofreq viterbi -f GCF_009858895.2_ASM985889v3_genomic.fna \
-o realigned_reads.bam deduped_align.bam
```

11.5 Indel Qualities

Calculating indel qualities is necessary for accurate variant calling. We will use the `lofreq indelqual` tool for this purpose:

Adjust the command below to calculate indel qualities, ensuring the reference genome and output file names are correctly specified:

`lofreq indelqual -f <reference_genome> -o <output_bam> <input_bam> --dindel`

The indel quality scores are added to the specified `output_bam` file.

```
lofreq indelqual -f GCF_009858895.2_ASM985889v3_genomic.fna \
-o indel_qual_report.bam realigned_reads.bam --dindel
```

11.6 Indexing Alignment

Indexing the alignment file is required for efficient access and processing. We will use `samtools index` to create the index.

You can sort the index with the following command: Replace '<input_bam>' with the filename of your BAM file that needs indexing:

`samtools index <input_bam>`

The index file is then created with the `.bai` extension.

Note. Before indexing is possible, something else needs to happen. Look back at earlier indexing steps and perform this step first.

```
samtools sort indel_qual_report.bam -o indel_qual_report_sorted.bam
samtools index indel_qual_report_sorted.bam
```

11.7 Variant Calling

Variant calling is a crucial step in the analysis of genomic data, particularly in the context of DNA sequencing. It involves identifying and cataloging genetic variations or variants, such as single nucleotide polymorphisms (SNPs), insertions, deletions, and structural variations, within an individual or a population's genome.

Common variant calling tools are GATK or Samtools, but in this tutorial lofreq is used. This is a variant calling tool that is commonly used for the identification of low-frequency variants in high-throughput sequencing data. It is designed to be particularly sensitive to variants present at low allele frequencies, making it useful for applications such as tumor heterogeneity analysis or the detection of rare variants.

Question: What is the role of the reference genome in the variant calling step? How does the choice of reference genome affect the interpretation of results? What if you would have a SARS-CoV-2 reference genome from later on in the pandemic?

> The reference genome serves as a template for aligning sequencing reads and identifying variations. The choice of reference genome is crucial as it directly impacts the interpretation of results. Using a reference genome from later in the pandemic may lead to different variant calls and interpretations compared to using an earlier reference genome. It is important to consider the evolutionary changes and dominant strains circulating at the time of the study when selecting an appropriate reference genome.

Construct the command by replacing placeholders with the appropriate filenames and parameters to perform variant calling:

```
lofreq call \
<input_bam> \
--ref <reference_genome> \
--call-indels \
--min-cov 50 \
--min-bq 30 \
--min-alt-bq 30 \
--min-mq 20 \
-o <output_vcf>
```

The called variants are written to the specified `output_vcf` file in the Variant Call Format (VCF).

```
lofreq call \
indel_qual_report.bam \
--ref GCF_009858895.2_ASM985889v3_genomic.fna \
--call-indels \
--min-cov 50 \
--min-bq 30 \
--min-alt-bq 30 \
--min-mq 20 \
-o variants.vcf
```

Question: What is the significance of the min-cov and min-bq parameters? How do these parameters affect the sensitivity and specificity of the variant calling?

The min-cov parameter sets the minimum read depth coverage required to consider a position for variant calling. Higher min-cov values increase specificity by reducing false positives but may miss low-frequency variants. The min-bq parameter sets the minimum base quality score required for a base to be considered in variant calling. Higher min-bq values increase specificity by filtering out low-quality bases but may reduce sensitivity. The choice of these parameters depends on the desired balance between sensitivity and specificity, sequencing depth, and the expected variant frequencies in the sample.

11.8 Annotating Variant Effects

Annotating variants helps understand their impact on the genome. We will use the `SnpEff` tool for annotating variant effects:

Replace '<genome_version>' and '<input_vcf>' with the appropriate identifiers and filenames:

`snpEff ann <genome_version> <input_vcf> -csvStats <output_csv> > <output_vcf>`

The annotated variants are written to the specified `output_vcf` file, and a CSV report is generated with the `-csvStats` option.

```
snpEff ann NC_045512.2 variants.vcf -csvStats report.csv \
> annotated_variants.vcf
```

Questions:

1. What is the most frequent variant type, and what percentage of the total variants does this constitute?
2. Not all mutations in the DNA lead to a difference on the protein level. How many of the mutations will have no functional impact? Is this more or less common than a mutation that changes the amino acid, and by how much?

1. From the snpEff ann output under the "Variants by type" section, it is noted that the most frequent variant type is SNP (Single Nucleotide Polymorphism). The report indicates that there are 162 SNPs, which constitute 92.57% of the total variants. These SNPs involve the substitution of one nucleotide for another, which can occur due to errors in DNA replication or due to mutagens.
2. In the "Effects by functional class" section:

```
Type , Count , Percent
MISSENSE , 283 , 71.64557%
NONSENSE , 31 , 7.848101%
SILENT , 81 , 20.506329%
```

   There are 81 silent mutations (20.51%) that have no functional impact on the protein level. Missense mutations, which change the amino acid, are more common than silent mutations by 51.14%, indicating that non-synonymous mutations are more than three times as common as synonymous mutations in this dataset.

11.9 Creating a Table of Variants

To create a tabular summary of the variants, we will use the `SnpSift` tool to extract relevant fields from the annotated VCF file:

Construct the command by replacing '<input_vcf>' with your annotated VCF filename:

```
SnpSift extractFields -e '.' <input_vcf> \
CHROM POS REF ALT QUAL DP AF SB DP4 \
'EFF[*].IMPACT' 'EFF[*].FUNCLASS' \
'EFF[*].EFFECT' 'EFF[*].GENE' 'EFF[*].CODON' \
> <output_tsv>
```

The extracted fields are written to the specified `output_tsv` file in a tab-separated format. *If you have issues running this command, try not to copy paste it but rather type it yourself.*

```
SnpSift extractFields -e '.' annotated_variants.vcf \
CHROM POS REF ALT QUAL DP AF SB DP4 \
'EFF[*].IMPACT' 'EFF[*].FUNCLASS' \
'EFF[*].EFFECT' 'EFF[*].GENE' 'EFF[*].CODON' \
> extracted_fields.tsv
```

11.10 Summarizing Data with MultiQC

MultiQC is a tool that aggregates results from various bioinformatics analyses into a single HTML report. We will use MultiQC to summarize the outputs from `fastp` and `Picard MarkDuplicates`:

Construct the command by replacing '<fastp_dir>' and '<picard_dir>' with the directories containing the output files from `fastp` and `Picard MarkDuplicates`, respectively:

```
multiqc -f -n <output_html> <fastp_dir>/ <picard_dir>/
```

```
multiqc -f -n multiqc_report.html fastp_output/ picard_output/
```

The MultiQC report is generated as the specified `output_html`, providing an overview of the quality control and duplicate removal steps.

Question: What is the purpose of a MultiQC report in the context of bioinformatics analysis?

A MultiQC report aggregates results from various bioinformatics analyses across multiple samples into a single interactive HTML report. This allows easier comparison of metrics between samples and tools, facilitating interpretation of large datasets. The report provides an overview of data quality, processing statistics, and potential issues, enabling researchers to assess the success of their analyses and troubleshoot problems.

Conclusion

- FASTQ Sanger version of the format is considered to be the standard form of FASTQ.
- Paired end data can be advantageous compared to single end data
- fastp and MultiQC are tools allowing to check the quality of FASTQ datasets. MultiQC is great to summarize results.
- The most common tools for mapping are Bowtie, BWA, BWA-MEM. You can use in-built genome to map against or upload one if it is missing.
- The standard format for storing aligned reads is SAM/BAM. The major toolsets to process these datasets are DeepTools, SAMtools, BAMtools and Picard.

By following this tutorial, you have gained hands-on experience with the basic workflow of NGS data analysis. We encourage you to explore further resources and experiment with different datasets to deepen your understanding of bioinformatics analysis.

Appendix: Description of file types

SAM/BAM file

The information contained in SAM and BAM files is identical. However, the SAM format is a text file, which can be read by a human, while a BAM file is a binary file format, which only makes sense to a computer. Mostly BAM files are used for long time storage, since they take up less space on a hard drive, and most software works more efficiently with them. Both SAM and BAM files can be sorted by chromosome and position, which makes it more efficient for a computer to search through it. For most tools, this is a requirement.

Example

```
<NAME> 83     1   69507   57 7S69M       =    69422    -153    <SEQ><QUAL><INFO>
<NAME> 99     1   69507   37   76M       =    69343     239    <SEQ><QUAL><INFO>
<NAME> 147    1   69507   57   76M       =    69398    -184    <SEQ><QUAL><INFO>
<NAME> 163    1   69508   37   76M       =    69613     180    <SEQ><QUAL><INFO>
<NAME> 133    1   69610   0      *       =    69610       0    <SEQ><QUAL><INFO>
<NAME> 113    1   532457  37   76M       =    451291 -81164    <SEQ><QUAL><INFO>
<NAME> 113    1   133406  37   76M       16   463967      0    <SEQ><QUAL><INFO>
```

Content

Col	Field	Description
1	QNAME	Query (pair) NAME
2	FLAG	bitwise FLAG
3	RNAME	Reference sequence NAME
4	POS	1-based leftmost POSition/coordinate of clipped sequence
5	MAPQ	MAPping Quality (Phred-scaled)
6	CIAGR	extended CIGAR string
7	MRNM Mate	Reference sequence NaMe ('=' if same as RNAME)
8	MPOS	1-based Mate POSistion
9	ISIZE	Inferred insert SIZE
10	SEQ	query SEQuence on the same strand as the reference
11	QUAL	query QUALity (ASCII-33 gives the Phred base quality)
12	OPT	variable OPTional fields in the format TAG:VTYPE:VALUE

VCF/BCF file

The relation between BCF and VCF is exactly the same as between BAM and SAM. However, the information contained within the file is different and explained below.

Example

```
#CHROM  POS       ID             REF          ALT     QUAL          FILTER
2     3721632     rs13390724       G           A     324.26          PASS
2     3721651     .                C           A     233.71          PASS
2     3721729     rs12105179       A           G     509.92          PASS
2     3724667     rs6723436        C           T      11.13          LowQual
```

```
INFO
AB=0.48;AC=1;AF=0.50;AN=2;DB;DP=21;MQ=56.62;MQ0=0;QD=16.87;SB=-0.01
AB=0.59;AC=1;AF=0.50;AN=2;DP=22;MQ=56.78;MQ0=0;QD=11.99;SB=-24.74
AB=0.62;AC=1;AF=0.50;AN=2;DB;DP=42;MQ=58.89;MQ0=0;QD=12.86;SB=-237.21
AB=0.00;AC=1;AF=0.50;AN=2;DB;DP=2;MQ=26.16;MQ0=1;QD=21.27;SB=-0.01

FORMAT  Sample
GT:AD:DP:GQ:PL  0/1:10,11:21:99:354,0,316
GT:AD:DP:GQ:PL  0/1:13,9:22:99:264,0,452
GT:AD:DP:GQ:PL  0/1:26,16:42:99:540,0,904
GT:AD:DP:GQ:PL  0/1:0,2:1:1.76:41,3,0
```

Content

INFO additional information: <key>=<data>[,data]

AA ancestral allele

AB allele balance

AC allele count in genotypes, for each ALT allele, in the same order as listed

AF allele frequency for each ALT allele in the same order as listed: use this when estimated from primary data, not called genotypes

AN total number of alleles in called genotypes

BQ RMS base quality at this position

CIGAR cigar string describing how to align an alternate allele to the reference allele

DB dbSNP membership

DP combined depth across samples, e.g. DP=154

MQ RMS mapping quality, e.g. MQ=52

MQ0 Number of MAPQ == 0 reads covering this record

NS Number of samples with data

QD Variant Confidence/Quality by Depth

SB strand bias at this position

FORMAT field

GT genotype; / : genotype unphased; | : genotype phased

DP read depth at this position for this sample

FT sample genotype filter indicating if this genotype was `"called"`

GL genotype likelihoods (log10-scaled) for all possible genotypes

GLE genotype likelihoods of heterogeneous ploidy

PL phred-scaled genotype likelihoods

GP phred-scaled genotype posterior probabilities

GQ conditional genotype quality, encoded as a phred quality $-10log_{10}P$(genotype call is wrong, conditioned on the site's being variant)

HQ haplotype qualities, two comma separated phred qualities

PS phase set

PQ phasing quality (phred-scaled probability)

EC list of expected alternate allele counts for each alternate allele

MQ RMS mapping quality