# Methods behind neoantigen prediction for personalized anticancer vaccines

Kiyana Godazandeh[†], Lies Van Olmen[†], Lore Van Oudenhove, Steve Lefever, Cedric Bogaert, and Bruno Fant[*]

*myNEO, Ghent, Belgium*

[*]*Corresponding author: e-mail address: bruno@myneo.me*

## Chapter outline

[†]These authors contributed equally to the manuscript.

## Abstract

Next to conventional cancer therapies, immunotherapies such as immune checkpoint inhibitors have broadened the cancer treatment landscape over the past decades. Recent advances in next generation sequencing and bioinformatics technologies have made it possible to identify a patient's own immunogenic neoantigens. These cancer neoantigens serve as important targets for personalized immunotherapy which has the benefit of being more active and effective in targeting cancer cells. This paper is a step-by-step guide discussing the different analyses and challenges encountered during in-silico neoantigen prediction. The protocol describes all the tools and steps required for the identification of immunogenic neoantigens.

## 1 Introduction

Cancer is characterized by an accumulation of somatic mutations. This can result in the development of abnormal cells that divide uncontrollably and have the potential to invade or spread to other parts of the body. The expression of these somatic mutations, however, subsequently generates cancer-specific actionable targets: neoantigens. These tumor-specific antigens bind to human leukocyte antigens (HLAs) molecules on the surface of cancer cells, enabling the immune system to recognize them as foreign epitopes leading to the destruction of the cancer cell. The unique array of neoantigens expressed by a patient's tumor cells has led the way to the development of personalized immunotherapy tailored specifically to an individual's tumor.

Identifying and targeting an individual patient's own specific immunogenic neoantigens to direct their immune system against the cancer cells has been a recent focus in the field of cancer immunotherapy. This personalized approach, as we will refer to it henceforth, can reap a more active and effective treatment: by comparing the genetic profile of a patient's tumor to their healthy tissue, mutations that result in the generation of neoantigens can be discovered. These neoantigens can then be targeted in a tailored approach by immunotherapy. There are currently two leading immunotherapeutic approaches which utilize neoantigens: cancer vaccination and T-cell therapy. Cancer vaccines can involve stimulating the patient's T-cells within the body so that they may recognize neoantigens, whereas T-cell therapy genetically engineers extracted T-cells from the patient so that, when they are reinfused, they are able to recognize specific neoantigens. It must be noted that the field is still young: the clinical successes that have been reported are still early involving phase I/II trials, hence personalized immunotherapy has yet to be approved. In the last few years, it has also become clear that neoantigen discovery faces several challenges impacting therapy efficacy and applicability. Indeed, since each individual patient can potentially express a considerable range of unique neoantigens with varying immunogenic potential, optimal prioritization becomes paramount to correctly identify the actionable ones, otherwise subpar therapies may be developed. This prioritization currently relies on bioinformatics methods. Here tumor-specific variations are detected from next-generation sequencing (NGS) data and the resulting neoantigens are inferred and prioritized in-silico based on various parameters (Lybaert et al., 2022).

Designing an individuals' personalized cancer treatment begins by comparing the NGS data of both the normal and tumor tissues of the patient to identify the tumor neoantigens. For this paper, both DNA and RNA sequencing is utilized. Whilst DNA sequencing (DNA-seq) of both the tumor and a control tissue will provide information about mutations that develop in the cancer cells, RNA sequencing (RNA-seq) will reveal which of these are actually transcribed. This combination can help create a comprehensive picture of the patient's tumor and its neoantigens, and therefore allows for a more precise and effective personalized cancer treatment approach (Conesa et al., 2016).

This paper describes a state-of-the-art bioinformatics neoantigen detection workflow comprised of four major steps: pre-processing and alignment, HLA-typing, variant calling and neoantigen prioritization. Once confidence in the individual steps has been built, the individual steps can be joined and automated using a workflow management tool to improve the efficiency and maintainability of the neoantigen discovery process. This is expanded upon further in Section 5.

The first step, **pre-processing**, is driven by quality control (QC) where errors are identified and, where possible, amended. These errors include the presence of adaptor sequences, poor-quality sequences at the start and end of the reads, polyA tails, and overrepresented reads. Sequence read trimming can be performed to correct some of these errors. It should be noted that trimming the reads too aggressively can lead to a shorter read length, and thus a loss of information, which can negatively impact the quality of mapping (Williams, Baccarella, Parrish, & Kim, 2016).

Once cleaned, **sequence alignment** to a reference genome can be performed to identify regions of the genome from which the reads originate. This step is performed independently for the DNA and RNA data. It is important to find an alignment tool that can both align reads that accommodate genomic variations such as mismatches, insertions and deletions (INDEL), and—in the case of RNA-alignment—contain non-contiguous genomic regions (Finotello, Rieder, Hackl, & Trajanoski, 2019). Various post-processing practices are also applied here to generate better quality reads including base quality score recalibration (BQSR) and, for RNA analysis, marking duplicates and splitting reads into exon segments. BQSR is performed to identify and correct any systematic errors in base quality scores generated by the sequencing machine. This can result in an over- or under-estimation of the base quality scores; correcting these errors typically results in more accurate SNV calling (Tian, Yan, Kalmbach, & Slager, 2016). It is important to mark duplicates in this process, as they identify multiple reads that match the same position on the genome. These can arise from PCR where reads are amplified to generate enough cDNA for sequencing (Parekh, Ziegenhain, Vieth, Enard, & Hellmann, 2016).

**HLA-typing** is required to identify which neoantigens are presented on the surface of cancer cells. The HLA molecules are the human form of major histocompatibility complex (MHC) molecules, the proteins on the surface of antigen presenting cells that bind the peptide antigens and present them for recognition by the T lymphocytes (Abbas, Lichtman, & Pillai, 2019). Different HLA molecules present different sets of neoantigens, therefore the likelihood of a neoantigen to be presented at the surface of a cell is determined by the HLAs expressed there (Finotello et al., 2019).

One of the most important steps in targeting neoantigens is **somatic variant calling**. For the detection of neoantigens, single nucleotide variants (SNVs) and indels are the most targeted form of mutations when carrying out neoantigen-directed therapies. As with the sequence alignment step, somatic variant calling is usually carried out independently for both RNA and DNA data to reliably identify expressed somatic variants from which neoantigens can be imputed for prioritization (Lang, Schrörs, Löwer, Türeci, & Sahin, 2022).

The last step in the process is **neoantigen prioritization**, where the predicted neoantigens are prioritized by their ability to be recognized by the T-cell receptor when they are bound to the HLA molecule. Ideally, many factors will have their role in neoantigen prioritization (Richters et al., 2019), however, here we will focus on two main factors: (1) the expression of the gene hosting the neoantigen mutation, which is determined using RNA-seq based gene expression analysis tools, and (2) the antigens' ability to bind HLA molecules, which is determined by peptide-HLA (p-HLA) binding affinity prediction tools (Lybaert et al., 2022).

The aim of this paper is to allow the user to understand and reproduce the major steps of neoantigen discovery for personalized immunotherapy. Here, we suggest publicly available, state-of-the-art tools, whilst also giving the user the freedom to deviate and expand, so as not to feel limited by following one constricting pipeline. The outputs of many of these steps can be recycled into pipelines not listed in this paper, for example for the identification of neoantigens derived from alternative splicing, transposable elements, and gene fusions, which is recommended for exploring more advanced personalized immunotherapy (Lang et al., 2022; Lybaert et al., 2022). For every tool used in the pipeline demonstrated below, there are numerous alternatives that can be explored to best match the user's samples and skillset. A comprehensive overview of such tools can be found in Lybaert et al.

## 2 Materials

### 2.1 Input data

#### *2.1.1 Samples*

This pipeline was created for the analysis of WGS data, but it can also be used with WES data with some small modifications.

Here, we use 3 WGS paired-end reads sequencing datasets:

1. 1 from normal DNA (30× coverage)
2. 1 from tumor DNA (90× coverage)
3. 1 from tumor RNA (30× coverage)

#### *2.1.2 Genome reference files*

Here is provided a table containing all the references used throughout this paper. Whenever a reference is needed as an input in the sections below, we recommend to follow the URL and download the specified file listed.

| Genome reference type | File name | Source name | URL |
|---|---|---|---|
| hg38 (GRCh38) reference genome | Homo_sapiens.GRCh38.dna.alt.fa.gz | Ensembl | https://ftp.ensembl.org/pub/release-108/fasta/homo_sapiens/dna/Homo_sapiens.GRCh38.dna.alt.fa.gz |
| hg38 (GRCh38) reference transcriptome | Homo_sapiens.GRCh38.cdna.all.fa.gz | Ensembl | https://ftp.ensembl.org/pub/release-108/fasta/homo_sapiens/cdna/Homo_sapiens.GRCh38.cdna.all.fa.gz |
| hg38 (GRCh38) peptide reference | Homo_sapiens.GRCh38.pep.all.fa.gz | Ensembl | https://ftp.ensembl.org/pub/release-108/fasta/homo_sapiens/pep/Homo_sapiens.GRCh38.pep.abinitio.fa.gz |
| hg38 (GrCh38) known variant sites for elPrep | 00-All.vcf.gz | dbSNP | https://ftp.ncbi.nih.gov/snp/organisms/human_9606/VCF/00-All.vcf.gz |
| hg38 (GrCh38) known variant sites- for GATK Mutect2 | 00-All.vcf.gz | dbSNP | https://ftp.ncbi.nih.gov/snp/organisms/human_9606/VCF/GATK/00-All.vcf.gz |
| hg38 (GrCh38) panel of normal | 1000g_pon.hg38.vcf.gz | GATK | https://storage.googleapis.com/gatk-best-practices/somatic-hg38/1000g_pon.hg38.vcf.gz |
| hg38 (GrCh38) germline population resource | af-only-gnomad.hg38.vcf.gz | GATK | https://storage.googleapis.com/gatk-best-practices/somatic-hg38/af-only-gnomad.hg38.vcf.gz |

*Continued*

—cont'd

| Genome reference type | File name | Source name | URL |
|---|---|---|---|
| hg38 (GrCh38) interval list | wgs_calling_regions.hg38.interval_list | GATK | https://console.cloud.google.com/storage/browser/_details/genomics-public-data/resources/broad/hg38/v0/wgs_calling_regions.hg38.interval_list |
| hg38 (GrCh38) annotated transcripts | Homo_sapiens.GRCh38.103.gtf.gz | Ensembl | http://ftp.ensembl.org/pub/release-103/gtf/homo_sapiens/Homo_sapiens.GRCh38.103.gtf.gz |
| hg38 (GrCh38) known cancer driver genes | Cancer Gene Census | COSMIC | https://cancer.sanger.ac.uk/census |

## 2.2 Hardware

Neoantigen discovery is best performed on a high-performance computing infrastructure as datasets can be quite large, hence may require significant computing power. A commercial or private cluster infrastructure is therefore recommended. In particular, the BQSR step described here requires 500 Gb RAM on average—any hardware solution considered should be mindful of this hard requirement.

- HPC infrastructure:
  - 25+ cores
  - 500+ Gb RAM
  - 1+ Tb disk space

## 2.3 Software

Below are listed the tools required for executing the protocol. It is recommended to download the latest version of each tool.

| Tool Name | Manual | Reference |
|---|---|---|
| FASTQC | https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/ | Andrews (2010) |
| Cutadapt | Required to run Trim Galore! | Martin (2011) |
| Trim Galore! | https://github.com/FelixKrueger/TrimGalore/blob/master/Docs/Trim_Galore_User_Guide.md | Krueger, James, Ewels, Afyounian, and Schuster-Boeckler (2021) |

—cont'd

| Tool Name | Manual | Reference |
|---|---|---|
| BWA | https://bio-bwa.sourceforge.net/bwa.shtml | Li (2013) |
| Samblaster | https://github.com/GregoryFaust/samblaster | Faust and Hall (2014) |
| GATK SortSam | https://gatk.broadinstitute.org/hc/en-us/articles/4418062801691-SortSam-Picard | McKenna et al. (2010), DePristo et al. (2011), van der Auwera et al. (2013) |
| elPrep | https://github.com/ExaScience/elprep | Herzeel et al. (2021) |
| GATK Mutect2 | https://gatk.broadinstitute.org/hc/en-us/articles/360036730411-Mutect2 | McKenna et al. (2010), DePristo et al. (2011), and van der Auwera et al. (2013) |
| GATK GetPileupSummaries | https://gatk.broadinstitute.org/hc/en-us/articles/360037593451-GetPileupSummaries | McKenna et al. (2010), DePristo et al. (2011), and van der Auwera et al. (2013) |
| GATK CalculateContamination | https://gatk.broadinstitute.org/hc/en-us/articles/360036888972-CalculateContamination | McKenna et al. (2010), DePristo et al. (2011), and van der Auwera et al. (2013) |
| GATK FilterMutectCalls | https://gatk.broadinstitute.org/hc/en-us/articles/360036856831-FilterMutectCalls | McKenna et al. (2010), DePristo et al. (2011), and van der Auwera et al. (2013) |
| STAR | https://github.com/alexdobin/STAR/blob/master/doc/STARmanual.pdf | Dobin et al. (2013) |
| GATK SplitNCigarReads | https://gatk.broadinstitute.org/hc/en-us/articles/360036858811-SplitNCigarReads | McKenna et al. (2010), DePristo et al. (2011), van der Auwera et al. (2013) |
| Strelka2 | https://github.com/Illumina/strelka/blob/v2.9.x/docs/userGuide/README.md | Kim et al. (2018) |
| bcftools | https://samtools.github.io/bcftools/bcftools.html | Danecek et al. (2021) |
| rtg tools | https://cdn.jsdelivr.net/gh/RealTimeGenomics/rtg-tools@master/installer/resources/tools/RTGOperationsManual/index.html | Cleary et al. (2014) |
| HLA-LA | https://github.com/DiltheyLab/HLA-LA | Dilthey et al. (2019) |

*Continued*

—cont'd

| Tool Name | Manual | Reference |
|---|---|---|
| Kallisto | https://pachterlab.github.io/kallisto/manual | Bray, Pimentel, Melsted, and Pachter (2016) |
| netMHCpan | https://services.healthtech.dtu.dk/service.php?NetMHCpan-4.1 | Reynisson, Alvarez, Paul, Peters, and Nielsen (2020) |
| netMHCIIpan | https://services.healthtech.dtu.dk/service.php?NetMHCIIpan-4.0 | Reynisson et al. (2020) |
| VEP | https://github.com/Ensembl/ensembl-vep | McLaren et al. (2016) |
| MuPeXI | https://github.com/ambj/MuPeXI/blob/master/doc/MuPeXI_User_Manual.md | Bjerregaard, Nielsen, Hadrup, Szallasi, and Eklund (2017) |

## 3 Methods

Each section of the methods segment is consistently organized in the following format: (1) a general description of the tool and the command used to execute the tool, (2) the input arguments followed by the files required to run the tool, (3) any additional parameters that are either required or recommended to run the tool with a short description as to what they do and (4) the output arguments (if needed to be specified) and the expected output when running the tool.

It is recommended to browse the manual of each of the tools to fully understand what each argument means and whether further optional arguments should be supplied prior to executing the command. To encourage this, the software table includes a manual column to direct the user to the current version of the manual for each of the tools.

### 3.1 Pre-processing

Pre-processing of sequencing data is an essential step to ensure that the samples are of acceptable quality for downstream processing. Pre-processing is run the same for both the DNA and RNA samples, and should be applied on each sample.

#### *3.1.1 Quality control using FASTQC*

FASTQC (Andrews, 2010) is a tool that assesses the quality of the inputted sequencing reads. FASTQC generates an overview of the analyzed file, as well as a variety of graphs which can be used to determine which sequences are of low quality and need to be removed prior to any further analysis. It is recommended to look at the FASTQC manual for a more detailed guide on both the basic arguments and an in-depth analysis to understand the output. In addition to cleaning unwanted

sequences from the reads, quality control is an important step in determining whether the analysis should be continued if the sample contains extreme anomalies such as an overall poor base quality or strong A/T C/G imbalance. To run FASTQC, execute the command *fastqc* on each of the samples with the following arguments:
**Input**:

- A **fastq.gz** file

**Output**:

- A **html** file containing the final reports which can be visualized using a web browser
- A **zip** file which contains the individual plots and data files from the report

### 3.1.2 Trimming using Trim Galore!

Once FASTQC has been completed, the results can be used to decide whether further steps are needed to trim the data in order to remove unwanted sequences. Trimming is the only *in silico* approach available to increase the quality of the starting sequencing data, and consists in two main strategies: adapter trimming and quality trimming. Adapter trimming is the removal of unwanted adapter sequences, DNA molecule pieces with constant known sequences used during the sequencing process. These adapters do not map to the human genome and will actively harm the mapping process of any read they are present on, and should therefore be removed. Quality trimming, on the other hand, removes the parts of the sequencing reads that have not reach a high enough base calling accuracy, and is based on Phred score analysis. A per-base PHRED score is given for each read in a FASTQ files, and reflects the possibility that the sequencer incorrectly calls a given base. A Phred Score of 20 indicates the likelihood of finding 1 incorrect base call among 100 bases, which is equivalent to a call precision of 99%. Lower base calling precision can starkly affect variant calling, and it is thus common practice to filter out bases with Phred scores lower than 20. As noted earlier, trimming the reads too aggressively can lead to read lengths too short to allow correct mapping to the transcriptome (Williams et al., 2016). It is also important to note that trimming may remove entire reads from the sequencing data; in other words, it reduces the effective size of the data available, to levels that may be too low to proceed. Should either of these situations arise, it could be advisable not to proceed with the rest of the analysis, opting instead to generate new data if possible.

There are many tools that can be applied for this step, here we opt for using Trim Galore! (Krueger et al., 2021) to enhance the quality of the sequencing data. What needs to be trimmed can be determined by looking at the FASTQC html files generated in Section 3.1.1 and executed with the assistance of the guidelines described in the Trim Galore! manual. The guidance here demonstrates how to run Trim Galore! with the default parameters, however these can be adjusted if needed by including further parameters. Trim Galore! is a wrapper script for FASTQC and Cutadapt therefore both of these tools need to be installed prior to running Trim Galore!.

To run Trim Galore! execute the command *trim_galore* on each of the samples with the following arguments:
**Input**:

- The **fastq.gz** files

**Parameters:**
- *–paired:* indicates the use of **paired FASTQ** files
- *–fastqc:* automatically performs FASTQC after trimming has been completed

**Output**:

- Trimmed **FASTQ** reads
- A Trim Galore! report
- A FASTQC **html** file containing the final reports which can be opened in a web browser. It is recommended to analyze this report to confirm that trimming has been correctly performed.

If further trimming is needed, this section should be repeated with adjusted parameters. Once quality control has been completed, the sequencing data can be used as input for the bioinformatics workflows that are described below: DNA analysis, RNA analysis, HLA typing, and neoantigen presentation.

## 3.2 DNA analysis

DNA analysis is performed on the pre-processed normal and tumor DNA paired-end sequencing files. It is recommended to process the normal and tumor DNA samples in separate directories. The RNA sample will be run later in Section 3.3. It is to be noted that while the tools used here are relatively robust and resistant to changes in tumor purity, the variant caller used on DNA is very sensitive to tumor contamination in the normal sample (Kim et al., 2018). It is therefore recommended to check the frequency of the normal_artifact filter in the final MuTect2 output; a high abundance of such calls may reflect normal contamination. In such case, it may be advisable to only proceed with the results of variant calling at the RNA level only; while less specific, they are indeed less affected by normal contamination (Kim et al., 2018).

### *3.2.1 Read alignment using BWA*

Read alignment is the first step of the DNA analysis workflow. BWA (Li, 2013) is the tool used to align the reads to the reference genome; more information about this tool and additional parameters can be found in its manual. BWA requires an indexed reference genome to be prepared, so that it can quickly and effectively retrieve reference sequence information from the genome during alignment. Indexing of the reference genome can be done using the BWA command *bwa index* where the reference genome is in a FASTA format. It is recommended to use the latest genome reference build, in this guide the hg38 reference obtained from Ensembl will be used.

Additionally, duplicates will be marked here using the tool samblaster (Faust & Hall, 2014), which can be added as an extension to the bwa command. The format of the code is different here, the samblaster command is piped into the bwa command. The *–M* argument must also be inserted to accommodate for the *bwa mem –M* option. Additionally, s*amtools view –Sb* is used to select an output to the sorted file in the binary BAM format, which is recommended to reduce the file size.

Once the reference has been indexed, run the BWA-MEM algorithm by executing the command *bwa mem* with the following arguments:
**Input**:

- The indexed **Homo_sapiens.GRCh38.dna.alt.fa.gz** reference genome
- The trimmed **FASTQ** files for the normal or tumor DNA paired-end reads from the output of Section 2.1.2

**Parameters**:

- *-M*: to mark shorter split reads as secondary hits. This parameter is required for samblaster as well as for compatibility with the downstream algorithms Picard and GATK
- *samblaster -M:* to mark duplicates, piped after the bwa command

**Output**:

- *samtools view –Sb -> sample.out.bam:* specified to output a **BAM** file containing the aligned reads

### 3.2.2 Sort the SAM file by coordinate using GATK SortSam

SortSam (DePristo et al., 2011; McKenna et al., 2010; van der Auwera et al., 2013) can be used to sort the SAM file from the output of Section 3.2.1 by coordinate, so that the reads are sorted by the order with which they appear on the reference genome. This tool requires many dependencies to run, which is why GATK suggests installing and activating the GATK and Samtools conda package managers. Run SortSam with Java using the command *java -jar picard.jar SortSam* with the following arguments:
**Input:**

- The **BAM** file with the aligned reads from Section 3.2.1, supplied using the parameter *-I*

**Parameters:**

- *-SORT_ORDER coordinate:* to sort the file by coordinate

**Output:**

- A **BAM** file containing the sorted reads

### 3.2.3 BQSR with elPrep

elPrep (Herzeel et al., 2021) is a high-performance tool, which among sequencing functions, can perform BQSR. elPrep relies on several tool-dependent file formats which are straightforward to generate: the reference genome in an elfasta file format and the known sites in an elsites file format. The elfasta reference file can be produced using the command "*elprep fasta-to-elfasta hg38.fasta hg38.elfasta*". The known variant sites file will contain the known polymorphic sites which will be excluded from the output. This file can be generated using the command "*elprep vcf-to-elsites dbsnp_138.hg38.vcf dbsnp_138.hg38.elsites*". Make sure to look at the elPrep manual for more information. To run BQSR, execute the command *elprep sfm* with the following arguments:

**Input**:

- The **BAM** file from Section 3.2.2 output
- *–reference*: The **hg38.elfasta** reference file
- *–known-sites:* The file **dbsnp_138.hg38.elsites** containing the known variant sites

**Parameters**:

- *–bqsr*: instructs elPrep to perform BQSR

**Output**:

- A **BAM** file containing the results of the BQSR-modified input file

### 3.2.4 Call Somatic SNV and INDEL variants using GATK MuTect2

The next stage is to call SNVs and INDELs using GATK MuTect2 (DePristo et al., 2011; McKenna et al., 2010; van der Auwera et al., 2013). For DNA variant calling, the "tumor-normal" mode of GATK MuTect2 should be used to process both the tumor and its matching normal sample. It is also recommended to supply files containing known variant sites to annotate such variants observed in the samples. These reference files provide a baseline for the SNV calling and both the gnomAD and PoN (Panel of Normals) reference files are provided by GATK.

gnomAD is a germline population resource which represents approximately 200 k exomes and approximately 160 k genomes; it provides a list of all variants identified in these samples which can be used here and can be used to calculate normal contamination which will be taken into account in **step 4** of this section. Additionally, GATK provides a list of variants identified in a publicly available PoN, which is comprised of 1000 normal samples. A custom panel can also be created using a minimum of 40 normal samples from young, healthy individuals with technically similar properties. This can be desirable when wanting to ensure that the properties of the PoN are as close as possible to the tumor sample investigated. More information on creating a PoN can be found in the GATK manual; in this protocol we will use the publicly available GATK PoN. An interval list is also required for this process, which is used to define positions of genomic regions in order to remove unwanted or unreliable areas to enable a faster analysis. An interval list for WGS data is supplied by GATK.

The first step is to call somatic SNVs and INDELs with Mutect2. This will generate an unfiltered Mutect2 call set which can then be filtered to ensure that only true somatic mutations will be retained

– Step 1: run the files through GATK MuTect2. To call somatic SNV and INDEL variants, run the command *gatk Mutect2* with the following arguments:

**Input**:

- *-I*: The **normal BAM** from the output of Section 3.2.3 (use the parameter -normal as demonstrated below to differentiate the normal sample from the tumor sample)
- *-I*: The **tumor BAM** from the output of Section 3.2.3
- *-R:* The **Homo_sapiens.GRCh38.dna.alt.fa.gz** reference genome
- *-pon:* The **1000g_pon.hg38.vcf.gz** PoN reference
- *–germline-resource:* The **af-only-gnomad.hg38.vcf.gz** gnomAD known germline variant list
- *-L:* The **wgs_calling_regions.hg38.interval_list** interval list for WGS

**Parameters:**

- *-normal:* supply the normal sample name with this parameter to differentiate the normal sample from the tumor sample

**Output**:

- *-O:* A **VCF** file containing all the variants identified in the sample
- A **.stats** file containing the variant statistics

The next steps are used to decide which of the mutation candidates from the VCF output of the previous step are likely to be true somatic mutations.

– Step 2: Run GetPileupSummaries. This is used to summarize read support for a set number of known variant sites. This is run on both the normal and tumor samples separately, and requires the input of the gnomAD variant list and the WGS interval file. Run the command *gatk GetPileupSummaries* with the following arguments:

**Input**:

- The **normal/tumor** output **BAM** file from **step 1**. Run these files separately
- *-V:* The **af-only-gnomad.hg38.vcf.gz** gnomAD known germline variant list
- *-L:* The **wgs_calling_regions.hg38.interval_list** interval list for WGS

Since the normal and tumor BAM files are run separately in this process, each run will generate its own output.

**Output**:

- A file called **normal_getpileupsummaries.table** and

- A file called **tumor_getpileupsummaries.table**

– Step 3: Run CalculateContamination to estimate cross-sample contamination in the reads from the tables from the output of **step 2**. This generates a table used in **step 4**, FilterMutectCalls, to estimate the fraction of reads obtained from cross-sample contamination for both normal and tumor samples. Run *gatk CalculateContamination* with the following arguments:

**Input**:

- *-matched:* **normal_getpileupsummaries.table**
- *-I:* **tumor_getpileupsummaries.table**

**Output**:

- The output is **calculatecontamination.table** which is a table providing the fraction contamination

– Step 4: Apply variant filtering using FilterMutectCalls. This applies the filters from Step 3 to the raw output of Mutect2 from step 1. Run *gatk FilterMutectCalls* with the following arguments:

**Input**:

- *-R:* The **Homo_sapiens.GRCh38.dna.alt.fa.gz** reference genome
- *-V:* The **VCF** file obtained from the output of **step 1** of this process
- *–stats:* The **.stats** file from the output of **step 1** of this process
- *–contamination-table:* The **calculatecontamination.table** table from step 3

**Output**:

- A filtered **VCF** file called **filtered.vcf**

In Section 3.4, variants present in both this file and the list generated later in Section 3.3.5 as part of the RNA analysis will be identified. This will then be used in Section 3.6.2 to predict p-HLA binding affinity and perform a MuPeXI-based peptide extraction.

## 3.3 RNA analysis

The RNA analysis is similar to the DNA analysis workflow described in Section 3.2, in that we aim to generate a list of SNV and INDELs. Different tools will be used for some of the processes due to efficiency and accuracy.

### *3.3.1 Read alignment with STAR*

For RNA-based read alignment in this protocol, STAR is used as it can accommodate for exons better (Dobin et al., 2013). More detailed information on other parameters that could be applicable for your sample can be found in the STAR manual. As with most aligners, STAR requires an indexed genome reference, which can be created using the Homo_sapiens.GRCh38.dna.alt.fa.gz reference genome and the file containing the annotated transcripts for the hg38 genome. To align the RNA reads, run *STAR* with the following arguments:

– Step 1: Create genome indices. The directory where the indices will be stored should be created before the STAR run with *mkdir* and should have writing permissions:

**Input**:

- *–genomeFastaFiles:* The **Homo_sapiens.GRCh38.dna.alt.fa.gz** reference genome
- *–sjdbGTFfile:* The **Homo_sapiens.GRCh38.103.gtf.gz** annotated transcripts

**Parameters**:

- *–runMode genomeGenerate:* to specify to STAR to create an index
- *–runThreadN:* specify the number of threads that will be used to generate the indices
- – *sjdbOverhang*: specify the length of genomic sequence around the annotated junction to be used. The default is 100.

**Output:**

- *–genomeDir*: The path to the directory created beforehand with the command *mkdir* where the indices will be stored
- Multiple output files named **prefix .run** containing the genome index

– Step 2: Align the RNA reads using STAR. For the *readFilesIn* parameter, make sure to input both trimmed paired end read files for the sample. To align the reads, execute the command *STAR* with the following arguments:

**Input**:

- *–genomeDir*: The path to the directory where the indices generated in **step 1** where stored
- *–readFilesIn:* The two trimmed paired-end read **FASTA** files for the tumor RNA samples from the output of Section 2.1.2, leaving a space between each name

**Parameters:**

- *–outSAMtype BAM SortedByCoordinate:* so that the output is a **BAM** file and the reads are sorted by coordinate
- *–runThreadN*: specify the number of threads that will be used to generate the indices

**Output**:

- An aligned **BAM** file

### 3.3.2 Identify duplicates with elPrep

In addition to BQSR (Section 3.3.4), elPrep (Herzeel et al., 2021) can be used to identify and mark duplicate reads in the aligned file to prevent overrepresentation of amplified regions. To identify the duplicates, execute the command *elprep sfm* with the following arguments:
**Input**:

- The **BAM** file from the STAR output in Section 3.3.1

**Parameters:**
- *–mark-duplicates:* to mark the duplicate reads

**Output**:

- The output of this process will yield a **BAM** file with its duplicates marked.

### 3.3.3 Split reads into exon segments with SplitNCigarReads

SplitNCigarReads (DePristo et al., 2011; McKenna et al., 2010; van der Auwera et al., 2013) is a tool used to split a read with N cigar into individual exon segments. This removes the Ns without removing any grouping information. It also performs hard clipping of any overhanging reads into the intronic regions and reassigns the mapping quality. To split the reads into exon segments, execute the command *gatk SplitNCigarReads* with the following arguments:
**Input**:

- *-R:* The **Homo_sapiens.GRCh38.dna.alt.fa.gz** reference genome
- The **BAM** files from the elPrep output in Section 3.3.2

**Output**:

- A **BAM** file containing reads split at N cigars.

### 3.3.4 BQSR with elPrep

It is a good idea afterwards to run BQSR with elPrep (Herzeel et al., 2021) as done previously in the DNA process. The parameters and input files should all be the same as with DNA BQSR, however the RNA files should be used as the input file.

### 3.3.5 Somatic SNV and INDEL calling using Strelka2

Like Mutect2, Strelka2 (Kim et al., 2018) is used to call somatic SNVs and INDELs in RNA-seq data. Strelka2 works in 2 stages, the first stage being configuration including adjusting parameters and inputting sample data, the second stage being workflow execution. It must be noted that Strelka2 requires at least Python version 2.6, and will not work with python versions 3.x. To ensure that Strelka can run smoothly using this Python version and to prevent conflicts with other software installations, it is recommended to run Strelka2 using a Python2-supporting docker such as provided by the Sanger institute (https://github.com/cancerit/strelka2-

manta), or in a virtual environment such as provided by conda—the strelka2 package being available on the bioconda channel. To run the configuration step of somatic SNV and indel calling using Strelka2, run *path_to_strelka/configureStrelkaSomatic-Workflow.py* with the following arguments:

– Step 1: Configuration

**Input**:

- *–normalBam:* The **normal DNA BAM** file from the BQSR output in Section 3.2.3
- *–tumorBam:* The **RNA BAM** file from the output of Section 3.3.4
- *–referenceFasta*: The **Homo_sapiens.GRCh38.dna.alt.fa.gz** reference genome

**Output**:

- A Python file called **runWorkflow.py**

– Step 2: Workflow execution. The next step is to execute the workflow stage of Strelka2, using the command *path_to_file/runWorkflow.py*. There are no inputs or parameters needed since all the configuration was already performed in step 1.

**Output**:

- A file called **somatic.snvs.vcf.gz** containing all the somatic SNVs in the tumor sample
- A file called **somatic.indels.vcf.gz** containing all the somatic INDELS in the tumor sample

In Section 3.2.4, variants present in both this output and the variant list generated as part of the DNA analysis (Section 3.2.4) will be identified. These will then be used in Section 3.6.2 to predict p-HLA binding affinity and perform a MuPeXI-based peptide extraction.

## 3.4 Obtaining a final variant list

### 3.4.1 Identify overlaps between DNA and RNA variants using bcftools isec

Once variants have been identified for both the DNA and RNA samples, the VCF files can be intersected to find variants present in both data types. This is to ensure that only expressed, and therefore actionable neoantigens are kept. This can be done using *bcftools isec* (Danecek et al., 2021). To save disk space, it is recommended to zip and index the VCF files. This can be done using rtg tools (Cleary et al., 2014). (*rtg bgzip* and *rtg index*). To identify the overlaps between DNA and RNA variants, execute the command *bcftools isec:*

**Input:**

- *-w1:* The DNA **filtered.vcf.gz** from the output of Section 3.2.4 and the RNA **somatic.snvs.vcf.gz** from the output of Section 3.3.5

**Output:**
- A **VCF** file containing variants unique to the **DNA VCF** file
- A **VCF** file containing variants unique to the **RNA VCF** file
- A **VCF** file containing the **overlapping variants** from both the DNA and RNA files

## 3.5 HLA-typing

To allow prediction of the p-HLA binding affinities, the HLA type of the sample needs to be determined. The HLA-I and HLA-II alleles can be predicted based on tumor DNA with the HLA-typing tool HLA-LA (Dilthey et al., 2019) by running./*HLA-LA.pl* with the following arguments.
**Input:**
- –*BAM*: The indexed **BAM** file generated with BQSR in Section 3.2.3 for the **tumor DNA** sample
- –*graph*: The indexed graph constructed of reference haplotypes (how to get the required data packages to build the graph is explained on the HLA-LA GitHub page)

**Parameters:**
- –*sampleID*: A variable containing a unique sample ID for each sample

**Output:**
The main output file of HLA-LA is a dataframe (**R1_bestguess.txt**) containing the specific HLA alleles and a number of quality indicators which represent the probability with which the HLA alleles have been called.

## 3.6 Neoantigen prioritization

In this section the information gathered in the previous sections is combined to predict the neoantigens. Based on the specific HLA types and the neoantigens derived from the called variants, the most important features of these neoantigens will be analyzed to select only the confident and actionable candidates.

### 3.6.1 Expression analysis with Kallisto

Analyze the expression of the host genes harboring the mutations with the RNA-seq based tool Kallisto (Bray et al., 2016) by running the *kallisto quant* command with the following arguments. This step is important because no or limited ($< 33$ TPM) (Wells et al., 2020) active transcription of the host gene reduces the likelihood of generating mutant peptides to be presented by HLA molecules.
**Input:**
- The trimmed paired-end **RNA FASTQ** files of Section 3.1.2
- –*index*: The indexed **Homo_sapiens.GRCh38.cdna.all.fa.gz** reference transcriptome, the transcriptome can be indexed with the *kallisto index* command.

**Output:**

- *–output-dir*: The directory to write the output to

The *Kallisto quant* command outputs different files in the output folder specified with the *–output-dir* argument. The output file that will be used in the following steps is the **abundance.tsv** file which contains for every transcript the length, the effective length, the estimated count and the abundance in transcripts per million (TPM).

### 3.6.2 p-HLA binding affinity prediction and peptide extraction with MuPeXI

In this step the p-HLA I and the p-HLA II binding affinities will be predicted with netMHCpan and netMHCIIpan (Reynisson et al., 2020), respectively, and peptides with predefined lengths around the missense variant mutations, indels and frameshifts will be extracted by means of MuPeXI (Bjerregaard et al., 2017). As opposed to most other similar software's, MuPeXI has the distinct advantage of screening each neoantigen against the complete human proteome to exclude potential self-peptides, which is why it is selected for this step. The binding affinity prediction steps with netMHCpan and netMHCIIpan are included in MuPeXI. To run MuPeXI, the following software and Python modules should be installed: Python 2.7, netMHCpan (HLA I) or netMHCIIpan (HLA II), Variant Effect Predictor (VEP) (McLaren et al., 2016), biopython (Cock et al., 2009), numpy (Harris et al., 2020) and pandas. Run MuPeXI with *path/to/MuPeXI.py* once for HLA I peptides and once for HLA II peptides with the following arguments:
**Input:**

- References:
  - cDNA: The **Homo_sapiens.GRCh38.cdna.all.fa.gz** reference transcriptome
  - pep: The **Homo_sapiens.GRCh38.pep.all.fa.gz** peptide reference
  - cosmic: The cancer gene census
- config.ini: this file will be automatically downloaded into the MuPeXI directory while downloading the MuPeXI repository. Instructions to fill this in can be found within the file and in the MuPeXI user manual on GitHub. The following paths should be specified in the config.ini file:
  - [netMHC]: the netMHCpan or netMHCIIpan binary path
  - [EnsemblVEP]: the binary path to VEP (VEP) and to the directory containing the cache database (VEPdir)
  - [References]: path to the used references
  - [PeptideMatch]: full path to the small C script named "pepmatch_db" downloaded with MuPeXI
  - *–vcf-file*: the **VCF** file from Section 3.4 containing the overlapping variants between the DNA and RNA tumor samples called by MuTect2 and Strelka2, respectively.
  - *–expression-file*: the **abundance.tsv** output from Section 3.6.1 containing for each transcript its transcript ID and mean expression.

**Parameters:**

- HLA I peptides:
  - Specify the HLA I binding affinity predictor netMHCpan in the **config.ini** file
  - –*alleles*: comma separated list of the specific HLA I alleles called with HLA-LA in Section 3.5. The format of the HLA alleles should be for example HLA-A02:01
  - –*length*: specify a peptide length from 9-11 amino acids
- HLA II peptides:
  - Specify the HLA II binding affinity predictor netMHCIIpan in the **config.ini** file
  - –*alleles*: comma separated list of the specific HLA II alleles called with HLA-LA in Section 3.5. The format of the HLA alleles should be for example HLA-DRB111:03
  - –*length*: specify a peptide length of 15 amino acids

**Output:**

The main output of MuPeXI is a **TSV** file with the extension .mupexi containing the extracted neoantigens (Mutant peptide column) ranked according to their priority score.

## 4 Concluding remarks

In this paper, we describe a state-of-the-art bioinformatics protocol for the prediction of potential neoantigens from DNA and RNA sequencing data. To run this entire workflow without the need of manual intervention, it can be built with a workflow management system like Snakemake (Section 5) (Köster & Rahmann, 2012). From the peptides predicted in this protocol, it is crucial to only select the neoantigens that are presented by the HLA molecules on the tumor's cell surface and are able to elicit an immune response—i.e., they need to be immunogenic (Lopes, Vandermeulen, & Préat, 2019; Lybaert et al., 2022).

Following this workflow, the immunogenic peptides can be selected based on the priority score obtained from the MuPeXI output. This priority score ranges from 0 (worst) to 1 (best) and is based on the product of p-HLA presentation, abundance term and a term related to dissimilarity between the mutant and its most similar normal peptide. Since the exact factors determining neoantigen immunogenicity remain unclear, this priority score should be considered as a guide until additional studies on neoantigen immunogenicity have been performed (Bjerregaard et al., 2017). The Tumor Neoantigen Selection Alliance (TESLA) has a defined set of features and corresponding cut-offs that determine the presentation likelihood and are associated with immunogenicity of HLA I neoantigens. This feature set is comprised of the expression level of the host gene ($>$ 33 TPM), the p-HLA I binding affinity ($<$ 34 nM) and the p-HLA I binding stability (half-life $>$ 1.4 h). Binding stability can be analyzed with netMHCstabpan (Rasmussen et al., 2016). It is to be noted that while this set of cut-offs was best available, it still produced a considerable amount of false negatives: only 55% of truly immunogenic peptides were labeled as such. For neoantigens meeting these criteria, the neoantigens with either low agretopicity or high foreignness are expected to be the most immunogenic with a recall rate of 75% (Kishton, Lynn, & Restifo, 2020; Wells et al., 2020). Immunogenicity prediction can be performed with tools like the IEDB immunogenicity predictor (open-source) or neoIM (Calis et al., 2013; Pfitzer, Lybaert, Bogaert, & Fant, 2022). To induce a broad and strong immune response while still limiting the chance of improper target selection and inclusion of false positive neoantigens, commonly up to 20 neoantigens are selected. However, the optimal number of neoantigens needed as target for personalized anticancer immunotherapy is still unknown (de Mattos-Arruda et al., 2020; Lopes et al., 2019; Lybaert et al., 2022).

The workflow presented here is focused on HLA I binding peptides derived from SNVs and INDELs. To make it more suitable for the detection of neoantigens in “cold tumors”—i.e., tumors with a low mutation level—it can be extended with the identification of non-canonical variants such as tumor-specific alternative splicing variants, gene fusions, and transposable elements (Capietto, Hoshyar, & Delamarre, 2022; Gupta, Li, Roszik, & Lizée, 2021; Lybaert et al., 2022). Additionally, although genomic data is incredibly resourceful for identifying potential neoantigens, further analyses using corresponding mass spectrometry data may be a strong asset in verifying the presence of the genomically identified neoantigens in the tumor cell ligandome (Xie, Shen, Gao, et al., 2023).

Although neoantigen prediction pipelines have already made promising progress, there is still a need for improvement. Most current approaches are limited to HLA I restricted neoantigens while the important role of HLA II peptides in cancer immunity becomes more and more clear (Blass & Ott, 2021; Fotakis, Trajanoski, & Rieder, 2021; Kishton et al., 2020; Sun, Chen, Meng, Wei, & Liu, 2017). Tools such as netMHCIIpan for the prediction of HLA II peptides exist, but with far lower performance. In addition, too little is currently known about the characteristics that make HLA II peptides immunogenic, negatively affecting the accuracy of these methods (Capietto et al., 2022; Gupta et al., 2021). In this protocol, netMHCpan is used to predict the presentation of neoantigens by HLA I molecules. A drawback of netMHCpan, and most HLA presentation prediction tools, is that their prediction is based merely on the p-HLA binding affinity. Although p-HLA binding affinity has a major influence on neoantigen presentation, there are many other steps in the process that strongly affect whether a neoantigen is actually presented. Other MHC-peptide binding and presenting tools such as neoMS, SHERPA, and NMER take other cellular mechanisms into account such as protein degradation and peptide transport, thereby greatly improving their performance (Gartner et al., 2021; Mill, Bogaert, van Criekinge, & Fant, 2022; Pyke et al., 2021). However, it should be noted that these tools are not open-source.

Furthermore, most neoantigen prediction pipelines predict immunogenicity only based on the interaction between the HLA molecule and the neoantigen. Even though it is an important feature of neoantigen immunogenicity, a strong p-HLA interaction alone is not sufficient for a neoantigen to be immunogenic. To induce an immune response, the p-HLA complex has to be recognized by the T-cell receptor causing the need for accurate computational methods to assess TCR-pHLA binding (de Mattos-Arruda et al., 2020; Lybaert et al., 2022). However, until such models prove to be more robust, the currently proposed protocol offers a simple and easy-to-use workflow for the identification and prioritization of neoantigens (Fotakis et al., 2021; Kishton et al., 2020).

## 5 Addendum

### 5.1 Workflow management systems: Snakemake

A bioinformatics protocol as discussed in this article involves the execution of a large number of chained steps, which can lead to increased complexity when running manually. Workflow management systems like Snakemake make it possible to automate these pipelines. A Snakemake workflow is defined in a 'Snakefile' composed of different rules, each representing a different step in the pipeline. The main structure of a rule consists of a rule name, the input and output files and the shell command or Python code to be executed to create the output from the input files. The input and output filenames may contain various wildcards whose values are automatically inferred by Snakemake from the available files. By using the output of a previous rule as input for the next rule, the steps are linked together, automating the protocol. When Snakemake encounters an error while running, the output is saved at the point of failure and output files created during the failing rule are deleted. Together with Snakemake's property to only execute rules if the output files are missing or if the input files are newer than the output, this prevents duplication of work when resuming the protocol. A Snakefile can be executed by using specific commands in the command line. More information about Snakemake and the commands to execute the Snakemake workflows can be found in the manual (https://snakemake.readthedocs.io/en/stable/).

### 5.2 Example Snakefile

This is an example Snakefile for executing the DNA pre-processing steps described in this protocol. It pre-processes DNA samples called normal_DNA_reads1.fastq.gz, normal_DNA_reads2.fastq.gz, tumor_DNA_reads1.fastq.gz and tumor_DNA_reads2.fastq.gz. This Snakefile can be executed by running the shell command.

```
snakemake -cores 1 normal_DNA_reads1_val_1.fq.gz
  tumor_DNA_reads1_val_1.fq.gz
normal_DNA_reads2_val_2.fq.gz tumor_DNA_reads2_val_2.fq.gz
normal_DNA_reads1_val_1_fastqc.htmltumor_DNA_reads1_val_1_fastqc.
  html
normal_DNA_reads2_val_2_fastqc.htmltumor_DNA_reads2_val_2_fastqc.
  html
```

```
rule fastqc:
  input:
    fastq1="{id}_reads1.fastq.gz",
    fastq2="{id}_reads2.fastq.gz"
  output:
    fastqc1="{id}_reads1_fastqc.html",
    fastqc2="{id}_reads2_fastqc.html"
  shell:
    '''
    fastqc {input.fastq1}
    '''
rule trim_galore:
   input:
    read1="{id}_reads1.fastq.gz",
    read2="{id}_reads2.fastq.gz"
   output:
    "{id}_reads1_val_1.fq.gz",
    "{id}_reads2_val_2.fq.gz",
    "{id}_reads1_val_1_fastqc.html",
    "{id}_reads2_val_2_fastqc.html"
   shell:
    '''
    trim_galore -paired -fastqc {input.read1} {input.read2}
    '''
```

## References

Abbas, A., Lichtman, A., & Pillai, S. (2019). *Basic immunology e-book: Functions and disorders of the immune system* (pp. 5–11). Elsevier.

Andrews, S. (2010). *FastQC: A quality control tool for high throughput sequence data*. http://www.Bioinformatics.Babraham.Ac.Uk/Projects/Fastqc.

Bjerregaard, A. M., Nielsen, M., Hadrup, S. R., Szallasi, Z., & Eklund, A. C. (2017). MuPeXI: Prediction of neo-epitopes from tumor sequencing data. *Cancer Immunology, Immunotherapy*, *66*(9), 1123–1130. https://doi.org/10.1007/S00262-017-2001-3.

Blass, E., & Ott, P. A. (2021). Advances in the development of personalized neoantigen-based therapeutic cancer vaccines. *Nature Reviews. Clinical Oncology*, *18*(4), 215–229. https://doi.org/10.1038/s41571-020-00460-2.

Bray, N. L., Pimentel, H., Melsted, P., & Pachter, L. (2016). Near-optimal probabilistic RNA-seq quantification. *Nature Biotechnology*, *34*(5), 525–527. https://doi.org/10.1038/nbt.3519.

Calis, J. J. A., Maybeno, M., Greenbaum, J. A., Weiskopf, D., de Silva, A. D., Sette, A., et al. (2013). Properties of MHC Class I presented peptides that enhance immunogenicity. *PLoS Computational Biology*, *9*(10), e1003266. https://doi.org/10.1371/JOURNAL.PCBI.1003266.

Capietto, A. H., Hoshyar, R., & Delamarre, L. (2022). Sources of cancer neoantigens beyond single-nucleotide variants. *International Journal of Molecular Sciences*, *23*(17). https://doi.org/10.3390/IJMS231710131.

Cleary, J. G., Braithwaite, R., Gaastra, K., Hilbush, B. S., Inglis, S., Irvine, S. A., et al. (2014). Joint variant and *de novo* mutation identification on pedigrees from high-throughput sequencing data. *Journal of Computational Biology*, *21*(6), 405–419. https://doi.org/10.1089/cmb.2014.0029.

Cock, PJ, Antao, T, Chang, JT, Chapman, BA, Cox, CJ, Dalke, A, et al. (2009). Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, *25*(11), 1422–1423. https://doi.org/10.1093/bioinformatics/btp163. PMID: 19304878, PMCID: PMC2682512.

Conesa, A., Madrigal, P., Tarazona, S., Gomez-Cabrero, D., Cervera, A., McPherson, A., et al. (2016). A survey of best practices for RNA-seq data analysis. *Genome Biology*, *17*(1), 13. https://doi.org/10.1186/s13059-016-0881-8.

Danecek, P., Bonfield, J. K., Liddle, J., Marshall, J., Ohan, V., Pollard, M. O., et al. (2021). Twelve years of SAMtools and BCFtools. *GigaScience*, *10*(2). https://doi.org/10.1093/gigascience/giab008.

de Mattos-Arruda, L., Vazquez, M., Finotello, F., Lepore, R., Porta, E., Hundal, J., et al. (2020). Neoantigen prediction and computational perspectives towards clinical benefit: Recommendations from the ESMO Precision Medicine Working Group. *Annals of Oncology*, *31*(8), 978–990. https://doi.org/10.1016/J.ANNONC.2020.05.008.

DePristo, M. A., Banks, E., Poplin, R., Garimella, K. V., Maguire, J. R., Hartl, C., et al. (2011). A framework for variation discovery and genotyping using next-generation DNA sequencing data. *Nature Genetics*, *43*(5), 491–498. https://doi.org/10.1038/ng.806.

Dilthey, A. T., Mentzer, A. J., Carapito, R., Cutland, C., Cereb, N., Madhi, S. A., et al. (2019). HLA*LA—HLA typing from linearly projected graph alignments. *Bioinformatics*, *35*(21), 4394–4396. https://doi.org/10.1093/BIOINFORMATICS/BTZ235.

Dobin, A., Davis, C. A., Schlesinger, F., Drenkow, J., Zaleski, C., Jha, S., et al. (2013). STAR: Ultrafast universal RNA-seq aligner. *Bioinformatics*, *29*(1), 15–21. https://doi.org/10.1093/BIOINFORMATICS/BTS635.

Faust, G., & Hall, I. M. (2014). *SAMBLASTER*: Fast duplicate marking and structural variant read extraction. *Bioinformatics*, *30*(17), 2503–2505. https://doi.org/10.1093/bioinformatics/btu314.

Finotello, F., Rieder, D., Hackl, H., & Trajanoski, Z. (2019). Next-generation computational tools for interrogating cancer immunity. *Nature Reviews Genetics*, *20*(12), 724–746. https://doi.org/10.1038/s41576-019-0166-7.

Fotakis, G., Trajanoski, Z., & Rieder, D. (2021). Computational cancer neoantigen prediction: Current status and recent advances. *Immuno-Oncology Technology*, *12*, 100052. https://doi.org/10.1016/J.IOTECH.2021.100052.

Gartner, J. J., Parkhurst, M. R., Gros, A., Tran, E., Jafferji, M. S., Copeland, A., et al. (2021). A machine learning model for ranking candidate HLA class I neoantigens based on known neoepitopes from multiple human tumor types. *Nature Cancer*, *2*(5), 563–574. https://doi.org/10.1038/s43018-021-00197-6.

Gupta, R. G., Li, F., Roszik, J., & Lizée, G. (2021). Exploiting tumor neoantigens to target cancer evolution: Current challenges and promising therapeutic approaches. *Cancer Discovery*, *11*(5), 1024–1039. https://doi.org/10.1158/2159-8290.CD-20-1575/666727/P/EXPLOITING-TUMOR-NEOANTIGENS-TO-TARGET-CANCER.

Harris, CR, Millman, KJ, van der Walt, SJ, Gommers, R, Virtanen, P, Cournapeau, D, et al. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2. PMID: 32939066; PMCID: PMC7759461.

Herzeel, C., Costanza, P., Decap, D., Fostier, J., Wuyts, R., & Verachtert, W. (2021). Multi-threaded variant calling in elPrep 5. *PLoS One*, *16*(2), e0244471. https://doi.org/10.1371/journal.pone.0244471.

Kim, S., Scheffler, K., Halpern, A. L., Bekritsky, M. A., Noh, E., Källberg, M., et al. (2018). Strelka2: Fast and accurate calling of germline and somatic variants. *Nature Methods*, *15*(8), 591–594. https://doi.org/10.1038/s41592-018-0051-x.

Kishton, R. J., Lynn, R. C., & Restifo, N. P. (2020). Strength in numbers: Identifying neoantigen targets for cancer immunotherapy. *Cell*, *183*(3), 591–593. https://doi.org/10.1016/J.CELL.2020.10.011.

Köster, J., & Rahmann, S. (2012). Snakemake—A scalable bioinformatics workflow engine. *Bioinformatics*, *28*(19), 2520–2522. https://doi.org/10.1093/BIOINFORMATICS/BTS480.

Krueger, F., James, F., Ewels, P., Afyounian, E., & Schuster-Boeckler, B. (2021). *FelixKrueger/TrimGalore: v0.6.7—DOI via Zenodo*. https://doi.org/10.5281/ZENODO.5127899.

Lang, F., Schrörs, B., Löwer, M., Türeci, Ö., & Sahin, U. (2022). Identification of neoantigens for individualized therapeutic cancer vaccines. *Nature Reviews. Drug Discovery*, *21*, 261–282. https://doi.org/10.1038/s41573-021-00387-y.

Li, H. (2013). *Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM*. http://arxiv.org/abs/1303.3997.

Lopes, A., Vandermeulen, G., & Préat, V. (2019). Cancer DNA vaccines: Current preclinical and clinical developments and future perspectives. *Journal of Experimental & Clinical Cancer Research*, *38*(1), 1–24. https://doi.org/10.1186/S13046-019-1154-7.

Lybaert, L., Lefever, S., Fant, B., Smits, E., de Geest, B., Breckpot, K., et al. (2022). Challenges in neoantigen-directed therapeutics. *Cancer Cell*. https://doi.org/10.1016/J.CCELL.2022.10.013.

Martin, M. (2011). Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet.Journal*, *17*(1), 10. https://doi.org/10.14806/ej.17.1.200.

McKenna, A., Hanna, M., Banks, E., Sivachenko, A., Cibulskis, K., Kernytsky, A., et al. (2010). The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Research*, *20*(9), 1297–1303. https://doi.org/10.1101/gr.107524.110.

McLaren, W., Gil, L., Hunt, S. E., Riat, H. S., Ritchie, G. R. S., Thormann, A., et al. (2016). The Ensembl variant effect predictor. *Genome Biology*, *17*(1), 1–14. https://doi.org/10.1186/S13059-016-0974-4/TABLES/8.

Mill, N. A., Bogaert, C., van Criekinge, W., & Fant, B. (2022). neoMS: Attention-based prediction of MHC-I epitope presentation. *BioRxiv*. https://doi.org/10.1101/2022.05.13.491845.

Parekh, S., Ziegenhain, C., Vieth, B., Enard, W., & Hellmann, I. (2016). The impact of amplification on differential expression analyses by RNA-seq. *Scientific Reports*, *6*(1), 25533. https://doi.org/10.1038/srep25533.

Pfitzer, L., Lybaert, L., Bogaert, C., & Fant, B. (2022). Improving T-cell mediated immunogenic epitope identification via machine learning: The neoIM model. *BioRxiv*. https://doi.org/10.1101/2022.06.03.494687.

Pyke, R. M., Mellacheruvu, D., Dea, S., Abbott, C. W., Zhang, S. V., Phillips, N. A., et al. (2021). Precision neoantigen discovery using large-scale immunopeptidomes and composite modeling of MHC peptide presentation. *Molecular & Cellular Proteomics: MCP*, *20*, 100111. https://doi.org/10.1016/J.MCPRO.2021.100111.

Rasmussen, M, Fenoy, E, Harndahl, M, Kristensen, AB, Nielsen, IK, Nielsen, M, et al. (2016). Pan-specific prediction of peptide-MHC class I complex stability, a correlate of T cell immunogenicity. *Journal of Immunology*, *197*(4), 1517–1524. https://doi.org/10.4049/jimmunol.1600582. PMID: 27402703; PMCID: PMC4976001.

Reynisson, B., Alvarez, B., Paul, S., Peters, B., & Nielsen, M. (2020). NetMHCpan-4.1 and NetMHCIIpan-4.0: Improved predictions of MHC antigen presentation by concurrent motif deconvolution and integration of MS MHC eluted ligand data. *Nucleic Acids Research*, *48*(W1), W449–W454. https://doi.org/10.1093/NAR/GKAA379.

Richters, M. M., Xia, H., Campbell, K. M., Gillanders, W. E., Griffith, O. L., & Griffith, M. (2019). Best practices for bioinformatic characterization of neoantigens for clinical utility. *Genome Medicine*, *11*(1), 56. https://doi.org/10.1186/s13073-019-0666-2.

Sun, Z., Chen, F., Meng, F., Wei, J., & Liu, B. (2017). MHC class II restricted neoantigen: A promising target in tumor immunotherapy. *Cancer Letters*, *392*, 17–25. https://doi.org/10.1016/J.CANLET.2016.12.039.

Tian, S., Yan, H., Kalmbach, M., & Slager, S. L. (2016). Impact of post-alignment processing in variant discovery from whole exome data. *BMC Bioinformatics*, *17*(1), 403. https://doi.org/10.1186/s12859-016-1279-z.

van der Auwera, G. A., Carneiro, M. O., Hartl, C., Poplin, R., del Angel, G., Levy-Moonshine, A., et al. (2013). From FastQ data to high confidence variant calls: The Genome Analysis Toolkit best practices pipeline. *Current Protocols in Bioinformatics*, *43*(1110). https://doi.org/10.1002/0471250953.BI1110S43.

Wells, D. K., van Buuren, M. M., Dang, K. K., Hubbard-Lucey, V. M., Sheehan, K. C. F., Campbell, K. M., et al. (2020). Key parameters of tumor epitope immunogenicity revealed through a consortium approach improve neoantigen prediction. *Cell*, *183*(3), 818–834.e13. https://doi.org/10.1016/J.CELL.2020.09.015.

Williams, C. R., Baccarella, A., Parrish, J. Z., & Kim, C. C. (2016). Trimming of sequence reads alters RNA-Seq gene expression estimates. *BMC Bioinformatics*, *25*(17), 103. https://doi.org/10.1186/s12859-016-0956-2. PMID: 26911985. PMCID: PMC4766705.

Xie, N., Shen, G., Gao, W., et al. (2023). Neoantigens: Promising targets for cancer therapy. *Signal Transduction and Targeted Therapy*, *8*, 9. https://doi.org/10.1038/s41392-022-01270-x.