# Implementing Bioinformatics Pipelines and User Interfaces for Selection of Immunotherapeutic Targets in Colorectal Cancer

**António Paulo**
antonio.do.paulo@ist.utl.pt
Instituto Superior Técnico
Lisbon, Portugal
December 2020

## ABSTRACT

Next-generation sequencing (NGS) allows the extraction of genomic information in a timely and efficient manner. Cancer treatment benefits from this, as tumor-specific mutations (neoantigens) leading to destruction of tumor cells by the immune system can be more easily identified resorting to bioinformatics pipelines. Pipelines that are able to process NGS data efficiently potentiate the timely development of treatments. In this work, we present a neoantigen identification bioinformatics pipeline capable of leveraging High-Performance Computing clusters. The pipeline covers all the steps required to turn NGS raw data from sequenced samples into ranked neoantigen predictions, while utilizing RNA data to verify whether the tumor-specific mutations are being expressed. The pipeline was developed as an overhaul to a previously used pipeline by an Immunogenomics research group, managing to reduce the execution time from days to hours, while keeping neoantigen prediction in line with previous results. The presented pipeline is open-source, freely available in GitHub [36, 50] and was designed with reproducibility, modularity and collaboration in mind. Additionally, this work includes a data visualization module through which users can visualize the pipelines' results (also available in GitHub [18]).

### Keywords

Bioinformatics pipeline; neoantigen identification; personalized cancer vaccine; next-generation sequencing; high-performance computing.

## INTRODUCTION

### Biological Fundamentals

Computational methods aid researchers in finding new therapeutics for cancer treatment. Next-Generation Sequencing (NGS) allows the sequencing of Deoxyribonucleic Acid (DNA) and Ribonucleic Acid (RNA) much faster and cheaper than was previously possible [5]. *Sequencing* is the process of determining the nucleotide order of DNA or RNA extracted from cells, which in our context are human cells. The entire set of an individual's DNA constitutes their *genome*. Inside our cells, DNA is transcribed into RNA, which in turn is translated into a protein. A portion of the genome that codes for a protein is known as a *gene*, and *alleles* are variations of a given gene.

Sequencing a human tissue sample using NGS technology can result in billions of *reads* [30, 37] – sequences of nucleotides each corresponding to some portion of the genome. Based on the sequencing data of several individuals, a human reference genome was created by the Genome Research Consortium (GRC) [19]. This reference genome is used by the scientific community as a point of comparison in NGS analyses. Any change in the DNA sequence relative to a reference genome is a *variant*. Variants in the coding part of a genome, i.e. genes, are known as coding variants. *Mutations* are variants that have deleterious or advantageous effects in cells and are rare events. In cancers, mutations on genes that control cellular growth often accumulate, leading to an uncontrolled proliferation of cancer cells. Variants in tumor cells that change protein-coding sequences may generate neoantigens. *Antigens* are substances that induce an immune response on our body, and *neoantigens* are antigens encoded by tumor-specific variants, i.e, variants not present in normal tissue. Neoantigens have been shown to play a significant role in mediating the destruction of tumor cells by the immune system [16, 17, 52]. Moreover, tumors with a higher number of tumor specific-mutations, and consequently more neoantigens have a better prognostic [32].

*T cells* are a type of white blood cells fundamental to the body's immune response. Tumor cells are killed through the action of Cytotoxic T Cells (CTLs) that recognize neoantigens that bind to the T Cell Receptor (TCR) and are presented by cell surface Major Histocompatibility Complex (MHC) molecules in tumor cells (see Figure 1). As such, numerous studies have targeted patient-specific neoantigens to clear tumors, with treatments using personalized neoantigen vaccines showing promising results [34, 44]. Note that the human leukocyte antigen (HLA) is equivalent to the MHC but is specific to humans.

### Bioinformatics Pipeline

To identify neoantigens, NGS data must go through several steps of analysis and transformation which can be accomplished using a bioinformatics *pipeline*: a series of software steps, usually chained together using a framework that con-

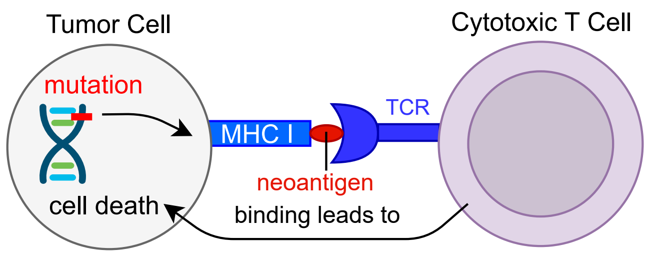


**Figure 1. Elimination of a tumor cell with a non-synonymous coding mutation by a CTL. This mutation originated a neoantigen that is then presented by a cell surface MHC molecule. The CTL then specifically recognizes the presented neoantigen and causes the tumor's cell death.**

trols the process flow. In complex pipelines the framework should ideally be a Workflow Management System (WFMS). A proper WFMS enables the pipeline to continue where it left off if interrupted, is able to utilize multiple computing nodes for added efficiency and handles job scheduling without requiring modifications to the existing code. Additionally, having native container support allows the reproducibility and portability of the pipeline. *Cromwell*, the WFMS used in our neoantigen identification pipeline, provides the aforementioned features. Figure 2 represents an example flow of a pipeline executing in a High-Performance Computing (HPC) cluster environment using a WFMS framework and software containerization

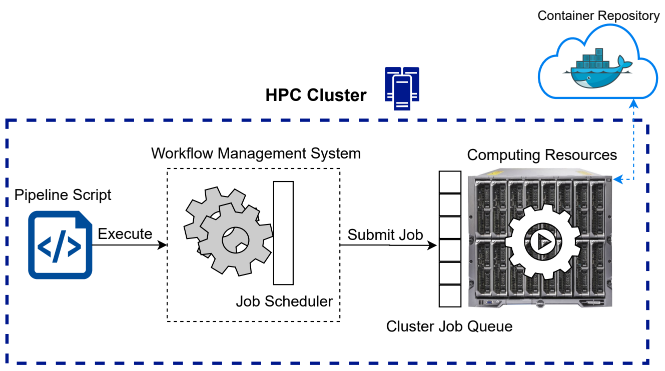


**Figure 2. Example process flow of a pipeline executing in an HPC cluster. A WFMS is used to create and submit the jobs to the cluster's queue without having had to explicitly write code in the pipeline script to do so. Moreover, containerization of the software, which provides reproducibility and portability to the pipeline by pulling container images from cloud repositories, is depicted in the upper right corner.**

Due to the complex nature of these pipelines in terms of the numerous inputs and transformations that the data goes through, it is valuable to present the processed data visually as it helps detecting errors such as malformed or mismatching inputs, or incorrect bioinformatics software configuration, thus avoiding drawing wrong conclusions about the processed data. With this in mind, our pipeline generates Quality Control (QC) plots and we implemented a separate R Shiny visualization module that provides a summary of the number and type of non-synonymous variants.

### Problem Formulation

In this work, we present a bioinformatics pipeline capable of processing a large amount of NGS data from patients and with it identify neoantigens that can be used by researchers to develop targeted cancer vaccines. The pipeline is destined (but not restricted) to analyze sequencing data from Colorectal Cancer (CRC) with low mutation numbers, with the key aspect being the identification of all tumor-specific variants that are coding *non-synonymous*, i.e., variants that cause changes to the proteins' sequence. This analysis was already being done, with an older pipeline, by the Immunogenomics group of the Department of Pathology of Leiden University Medical Center (LUMC) [28], environment where and for which the new pipeline was developed. The main issue with the previous (makefile [21]) pipeline was that its execution time was in the order of days, something that was vastly improved with the new pipeline. Contrary to the previous pipeline, the new pipeline is capable of leveraging the computational power of High-Performance Computing (HPC) clusters, including the one available at LUMC.

## TYPICAL NEOANTIGEN IDENTIFICATION WORKFLOW

Figure 3 summarizes the typical neoantigen identification workflow [24]. Firstly, tumor and normal (healthy) tissue have their DNA extracted (also RNA for the tumor) which is sequenced using NGS technology. Afterwards, QC, adapter trimming and read mapping are performed. In read mapping, millions of independent nucleotide sequences (reads) are mapped to the corresponding position in the reference genome. After the reads are aligned, variant calling is performed, i.e., nucleotide differences between the tumor and normal reads are identified. Using the RNA mapping information, it is possible to verify whether a variant is being expressed. Separately, the patient's HLA type is identified. Finally, binding prediction to the patient's HLA type is done for all tumor-specific variants that are present in the RNA sequences.

## DEVELOPED NEOANTIGEN IDENTIFICATION PIPELINE

We aimed to build an *integrated pipeline*, i.e., a pipeline comprised of several modules that runs seamlessly after issuing a simple command from a Command-line Interface (CLI). A *module* represents a part of the typical neoantigen detection workflow. The pipeline consists of six modules: the first three pipeline modules are integrated, whilst the remaining three are not. As such, the latter have to be run separately as standalone CLI tools, not benefiting from the Cromwell WFMS (detailed later). Furthermore, at LUMC there is an HPC Cluster where users can submit computational jobs that get queued. The pipeline was developed in this HPC cluster, but the implementation is not bound to it.

### Overview

The pipeline's main goal is to identify neoantigens starting from tumor-normal matched tissue samples, i.e., tumor and normal tissue samples from a given cancer patient. Multiple matched samples from different patients can be processed by the integrated pipeline in a single run. Figure 4 summarizes the developed pipeline, including the steps that precede running it in an HPC cluster.

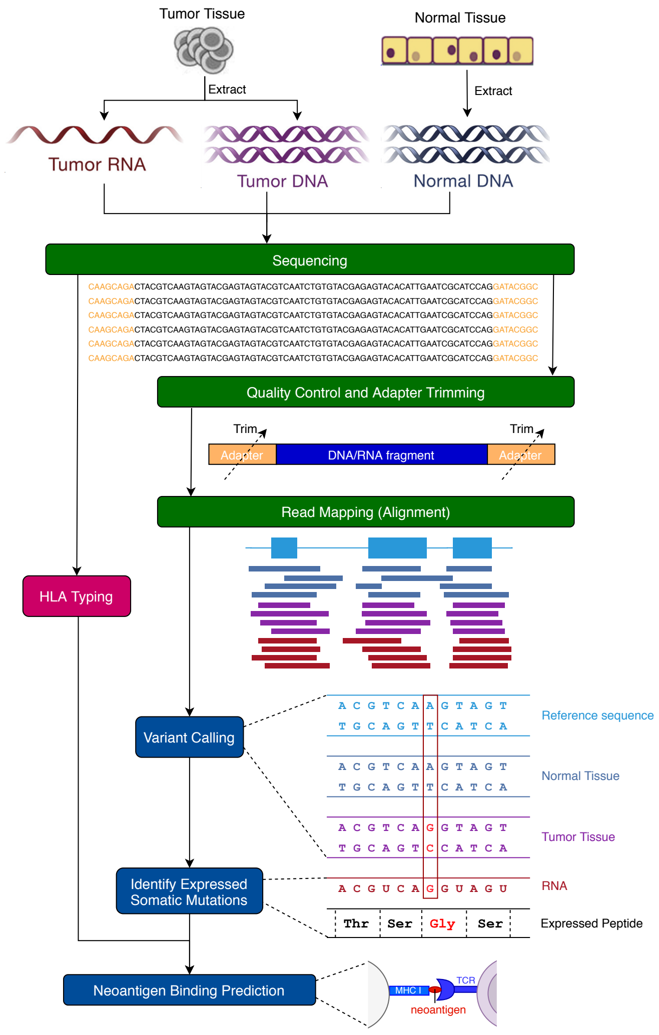


**Figure 3. Typical neoantigen identification workflow.**

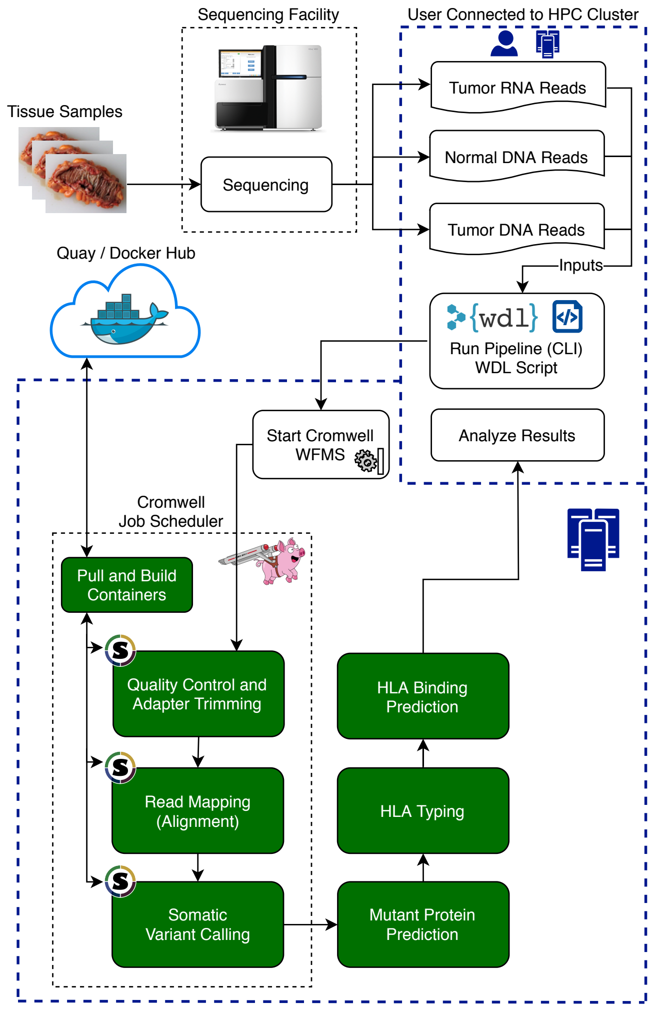


**Figure 4. Neoantigen identification pipeline in the context of LUMC's Cancer Immunogenomics research group which has access to an HPC cluster. The pipeline's steps are highlighted in green, and inside the blue dotted line is what concerns the HPC cluster environment where the pipeline is run.**

Before running the pipeline, tumor-normal matched tissue samples are collected and sent to a sequencing facility that sends back the files with the reads pertaining to the samples. These reads are some of the pipeline's inputs. Afterwards, while the user is connected to the HPC cluster (where the Cromwell WFMS is installed), the pipeline is executed by running its main script, written in the Workflow Description Language (WDL) (detailed later). This triggers Cromwell which is responsible for job scheduling. Cromwell decides – based on the computing resources required and the dependencies between tasks – and submits jobs to the cluster that correspond to modules' tasks. The first three modules of the pipeline – QC and Adapter Trimming, Read Mapping, and Somatic Variant Calling – pull container images from the Docker Hub [13] or Quay [39] container repositories, and then build Singularity container images. Pulling and building are handled by the Singularity container tool [45]. Singularity is used due to the safety concerns of HPC clusters. The remaining three pipeline modules – Mutant Protein Prediction, HLA Typing and HLA Binding Prediction – are run separately as they are not yet implemented using WDL. Below, we detail each pipeline module and the surrounding development aspects.

## Workflow Description Language

The pipeline was implemented using WDL [51], a workflow building language designed to be accessible and easily understandable. WDL allows to specify *workflows* and *tasks*. Workflows call tasks (or other workflows), allowing to chain together software execution. Moreover, WDL has a simple to use mechanism called *scattering*. By wrapping a certain portion of code into a scatter block, the workload is parallelized, allowing for more efficient computation. Additionally, WDL provides a practical *File* primitive that abstracts file paths into objects, removing the extra complexity of working with file paths when compared to the standard string primitive. WDL can be used to write workflows that transparently use container technologies such as Docker, and is suited for describing large-scale workflows in HPC clusters and cloud environments where tasks are scheduled in parallel across numerous nodes.

### Cromwell WFMS

WDL workflows can be run using Cromwell, a WFMS geared towards scientific workflows whose development is linked to WDL. Cromwell manages job submission into computing clusters and scheduling without pipeline developers having to write code for that. Furthermore, Cromwell has a caching system that avoids re-running previously executed tasks, avoiding the entire pipeline to run when only part of it (e.g., user-defined inputs) changed. Before running pipelines using Cromwell, it needs to be configured. The configuration depends on the execution environment, which in our case was an HPC with a Sun-Grid Engine backend.

### Singularity Containers

Containers address the important aspect of building a reproducible and portable pipeline, allowing different users to have the same software configuration simply by linking to the same container image. By default, all tasks point to a tested container image. Singularity was the chosen container tool because, due to its stricter security model, it is usually adopted in HPC environments. Although the WDL scripts link to Docker images, Cromwell can be configured to use Singularity to pull and convert Docker images prior to running a given task.

## INTEGRATED PIPELINE

### Quality Control and Adapter Trimming

The first step is scattering the files resulting from sequencing the tumor and normal tissue samples. These files can be multiple gigabytes in size. By splitting them into smaller files, the "QC" WDL workflow jobs can be called in parallel. Following the guidelines, quality control is run twice (FastQC tool [3]), before and after trimming adapter sequences (not part of the patient DNA) with Cutadapt [33]. Finally, by running the MultiQC [15] tool, FastQC reports and other metrics are aggregated and plotted into a single HTML file. This file addresses part of the result visualization concerns mentioned previously by giving the user insights into the quality of the reads (which impact the quality of the downstream results).

### Read Mapping (Alignment)

Reads are mapped using BWA-MEM [29], the Burrows-Wheeler Alignment (BWA) tool used to align reads ranging from 70 base pairs (bps) to a few megabases (ours being over 150 bps). Succinctly, after the scattered sequencing files are processed by the QC and Adapter Trimming module, they are aligned against a (user-defined) reference genome, resulting in Binary Alignment Map (BAM) files. Then, using Picard [11], duplicate reads (originating from a single DNA fragment) in the BAM files are located and tagged. With the Genome Analysis Toolkit (GATK) [9], the (scattered) BAM files go through base quality score recalibration. This step detects systematic errors made by the sequencing machine when it estimates the accuracy of each base call [8]. Additionally, BAM metrics are collected and later gathered by MultiQC, and the scattered BAM files gathered into one (for each sample).

### Somatic Variant Calling

Depending on user choice, up to three different variant callers can be run simultaneously and their results combined: Mutect2 [6], Strelka2 [25] and VarDict [27]. The advantage of having three variant callers is maximizing the number of true protein changing variants called. Generally speaking, there are three types of variants: Single Nucleotide Variants (SNV), insertion or deletion (indel) and Structural Variants (SV). Not all variant callers call all three variants types because fundamentally different algorithms are required [53]. Table 1 summarizes the types of variants called by each of the variant callers used. This table also contains a column for Multi-Nucleotide Variants (MNV), i.e., SNVs that are located together. Providing this combination of variant callers is a benefit of using our pipeline when compared to other neoantigen identification pipelines that restrict their results to a certain type of variant (e.g, the Epi-Seq pipeline [14] only handles SNVs).

| | SNV | MNV | Indel | SV |
|---|---|---|---|---|
| **Mutect2** | Yes | Yes | Yes | No |
| **Strelka2** | Yes | No | Yes | Yes |
| **VarDict** | Yes | Yes | Yes | Yes |

**Table 1. Variant types called by Mutect2, Strelka2 and VarDict**

In an independent benchmark [55], the sensitivity (true positive rate) and specificity (true negative rate) of well-established somatic variant callers such as Mutect and Strelka were compared. Strelka and Mutect were found to have achieved significantly higher sensitivity at the lowest Variant Allele Frequency (VAF), i.e., the minimum frequency to consider a variant true, with similar or lower false positive rates than the other variant callers. Furthermore, the previous pipeline at LUMC used Mutect and Strelka as well. Although the versions were older, we verified that the results between the new and the previous pipelines were identical. VarDict is capable of calling every type of variant. Additionally, in another benchmark [54], it was shown to achieve good accuracy, provided the VAF threshold is carefully chosen.

As a notable addition to the Mutect2 workflow when compared to the previous pipeline we highlight the panel of normals workflow [10]. The latter improves variant filtering using a database with variants found in the normal tissue of other patients. The logic is that, if a candidate somatic variant was detected in a given patient but it was also present in the panel of normals database, then it is likely to be a normal variant and can thus be filtered out.

### Modularity

The pipeline's source code [50], comprises several Git submodules [20]. Essentially, Git submodules are Git repositories that are kept as subdirectories of another Git repository. The pipeline is comprised by the following Git submodules:

- "QC": responsible for adapter trimming and gathering QC metrics

- "BamMetrics": gathers metrics from read alignment files.

- “gatk-preprocess”: produces a report on the observed base quality scores and recalibrates the read alignment files based on the report.
- “somatic-variantcalling”: responsible for running the somatic variant callers
- “tasks”: collection of reusable tasks called by the other workflows. It includes, for example, read mapping tasks.

The division into Git submodules enables integrating them easily into other WDL pipelines. All submodules are dependent on the “tasks” submodule. Otherwise, they are mostly independent and code changes should not involve rewriting multiple parts of the pipeline. Located at the root of the repository we find the main script, “pipeline.wdl” which runs the entire pipeline.

### Continuous Integration and Testing

For Continuous Integration (CI) purposes - the practice of merging in small code changes frequently, rather than merging in a large change at the end of a development cycle - the Travis CI [48] platform is used. Although normally requiring a paid license, it is free for open source projects such as our integrated pipeline. Travis CI automatically builds and tests code changes, providing immediate feedback on whether the changes were successful. The pytest framework [26] was used to write the integration tests that are run by Travis CI. The tests use small and well-described input data to verify that the pipeline runs successfully and whether the outputs change after new code is pushed to the repository. For example, to test the Somatic Variant Calling module, inputs smaller than 1 Megabyte containing NGS reads are used to confirm that the variants called remain the same after new code is pushed to the repository.

## STANDALONE PIPELINE MODULES

### Mutant Protein Prediction

Isovar [42] is used to predict the mutant protein sequences from the identified variants. It uses RNA sequencing reads to assemble the most abundant coding sequence for each mutation. One of the advantages of using RNA to determine the coding sequence for mutations is the phasing of somatic and germline (present both in tumor and normal cells) variants. If phasing is not taken into account, the resulting protein sequences may not match the ones produced by tumor cells, thus being an incorrect choice to include in a tumor vaccine. Figure 5 presents an overview of Isovar, showing the phasing of somatic and germline mutations.

### HLA Typing

HLA typing is crucial in neoantigen binding prediction to be able to assess how strongly variant peptides bind to the patient’s HLA molecule. Strong binders are more likely to be recognized by the immune system (see Figure 1). The OptiType [46] tool is regarded as the most accurate for HLA class I typing [4]. OptiType takes the sequencing reads from a given patient and maps them against all HLA class I alleles, allowing it to predict the optimal HLA alleles.

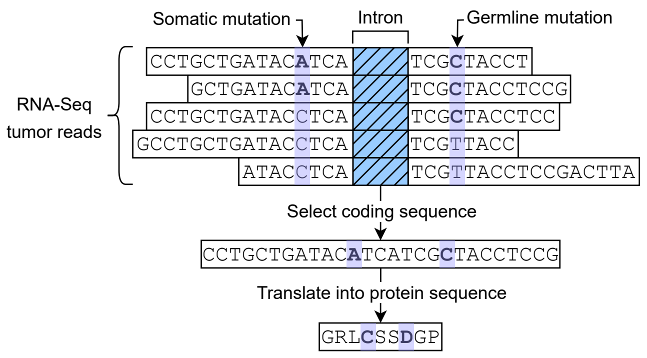


**Figure 5. Isovar tool overview including somatic and germline mutations phasing.**

### HLA Binding Prediction

For the HLA binding prediction module we developed a Python tool [36]. It depends on a modified version of mhctools [41], a Python Application Programming Interface (API) to commonly used MHC binding predictors. We use mhctools to run NetMHC [2] and NetMHCpan [40] as they are well-established, showing great performance in HLA class I binding prediction [22]. We extended mhctools [43] to expose a type of peptide ranking provided by NetMHCpan that is not made available by mhctools. Succinctly, the added functionality differs in the data that was used to train the NetMHCpan predictor, consequently affecting how peptides are ranked.

Our python tool, first takes the mutant protein sequences from Isovar alongside the respective non-mutant sequences (normal sample) and aligns them against one another (global pairwise alignment, exemplified in Listing 1). This allows to locate the mutations in the sequences. Then, small peptides with (user-defined) sizes are extracted around the mutations. Using BLAST+ [12], these small peptides are queried against a user-defined human protein database, such as Swiss-Prot [1] filtered to only contain human sequences. As a novel approach, we filter out peptides that matched completely to those on the database. The rationale is that peptides that are naturally found on humans are not good candidates to be included in a cancer fighting treatment. Finally, using our modified version of the mhctools API, we run netMHC and netMHCpan to predict the binding of the short peptides that remained after filtering (with our novel approach), to the patient’s HLA alleles. The main output is comprised of these binding predictions, where the indication of binding strength (weak, strong or no-binding) appear.

**Listing 1. Global pairwise alignment of two protein sequences. On the top is the non-mutant and on the bottom the mutant sequence. A 6 nucleotide deletion occurred (indel variant), leading to a 2 amino-acid gap in the middle of the mutant sequence.**

```
KIHVTPLIPGKSQSVSVSGPGPGPGPGLCPGPNVLLNQQNPPAPQPQHL--
|||||||||||||||||||||||||  ||||||||||||||||||||||
KIHVTPLIPGKSQSVSVSGPGPGPG--LCPGPNVLLNQQNPPAPQPQHLRP
```

Additionally, our tool provides two other types of outputs: 1) a log containing information such as the resulting global pairwise alignments and the tool’s execution progress. 2) Two files useful for laboratory work: one for mass spectrometry and another for peptide reactivity testing. Mass spectrometry can be used to identify peptides presented by MHC molecules (refer back to Figure 1). The mass spectrometry file consists of variant identifiers that are generated based on the mutation position with the respective mutant sequence alongside it. Succinctly, peptide reactivity testing checks for a neoantigen-specific response of T cells. The peptide reactivity testing file contains variant identifiers followed by the mutant sequences of size 25 that are built around the respective mutation position. These two files are not part of the typical neoantigen identification workflow. However, since they are helpful in facilitating the LUMC Immunogenomics research group workflow, they can also be valuable for other research groups doing a similar type of laboratory work.

## RESULT VISUALIZATION

Besides the aforementioned QC plots generated by the pipeline, we developed a separate R Shiny application. Shiny [38] is an R package to build interactive web applications with a focus on simplicity of development and redistribution, responsiveness in the interaction with the application for the end user and appealing default User Interfaces (UI) based on Bootstrap [7], a widely used front-end component library. The UI can be written in R but also directly using the common web development languages – HTML, CSS and JavaScript. Our Shiny application's source code is available in GitHub [18] and hosted online in the Shinyapps cloud [35] (experimental). In this application, we generate two bar plots using an R data scraping script that gathers and processes the data needed. The plots show the number of protein changing variants per sample and whether they are expressed (Figure 6), and the type of variants (Figure 7).

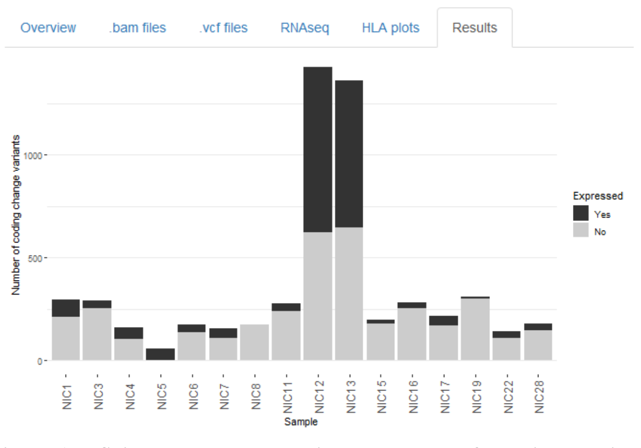


**Figure 6. R Shiny module plot showing the number of protein changing variants per sample, while indicating whether the variants are expressed (black) or not expressed (grey).**

## VALIDATION

### Synthetic Data

To validate variant calling performance, we ran the Somatic Variant Calling module of our pipeline on a paired tumor-normal synthetic sample from the ICGC-TCGA DREAM Mutation Calling Challenge that took place in 2013 [23], a commonly used dataset for the assessment of somatic variant callers. The whole genome sequencing reads were aligned to the GRC Human Build 37 (GRCh37) reference. Moreover, for each simulated tumor, a file containing the ground truth (computationally added) variants was provided. We focused solely on SNVs because these were consistently represented across all variant callers and the ground truth file.

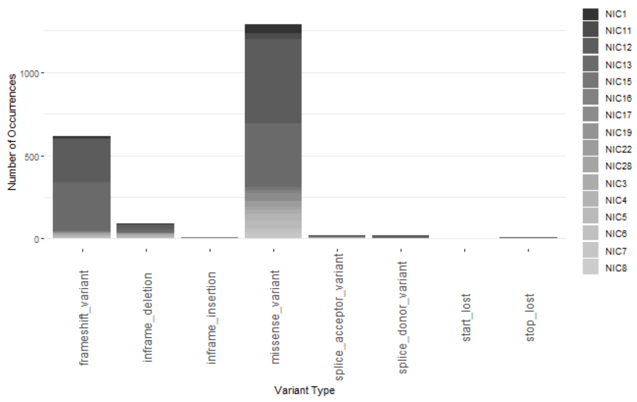


**Figure 7. R Shiny module plot showing the number of occurrences for a set of variant types. The samples' identifiers are shown on the right.**

$$Sensitivity = \frac{TP}{Positives} = \frac{TP}{TP + FN} \quad (1)$$

$$Specificity = \frac{TN}{Negatives} = \frac{TN}{TN + FP} \quad (2)$$

| | Mutect2 | Strelka2 | VarDict |
|---|---|---|---|
| **Sensitivity** | 0.973 | 0.957 | 0.987 |
| **Specificity** | 0.981 | 0.995 | 0.791 |

**Table 2. Mutect2, Strelka2 and VarDict sensitivity and specificity with synthetic data.**

| | $M2 \cup S2 \cup VD$ | $M2 \cup S2$ |
|---|---|---|
| **Sensitivity** | 0.991 | 0.980 |
| **Specificity** | 0.839 | 0.992 |

**Table 3. Mutect2 (M2), Strelka2 (S2) and VarDict (VD) unions' sensitivity and specificity with synthetic data.**

Recognizing that we are before a binary classification test, a variant called by our pipeline is considered either true or false, depending on whether it is in the ground truth file. Thus, each variant called can be classified as a True Positive (TP), False Positive (FP), False Negative (FN), or True Negative (TN).

In our case, a TP is a variant call that is also found in the ground truth file; a FP is a variant call that was not in the ground truth file; a FN is a variant that is in the ground truth file but that was either not called at all (missed) or that was called but was filtered by the variant caller; a TN is a variant call that was filtered by the caller and that is not in the ground truth file. Consequently, the number of TP plus FN totals the number of variants in the ground truth file, which is 3537.

| Sample | Target Variants | Mutect2 | Strelka2 | VarDict | Intersection | Require Inspection | Filtered by All |
|---|---|---|---|---|---|---|---|
| NIC3 | 19 | 19 | 19 | 18 | 18 | 1* | 0 |
| NIC4 | 30 | 29 | 30 | 30 | 29 | 1 | 0 |
| NIC5 | 50 | 46 | 49 | 50 | 45 | 5 | 0 |
| NIC6 | 23 | 22 | 23 | 23 | 22 | 1 | 0 |
| NIC7 | 33 | 27 | 29 | 29 | 27 | 3* | 3** |
| **Total** | 155 (100%) | 143 (94%) | 150 (96.8%) | 150 (96.8%) | 141 (91%) | 11 (7.1%) | 3 (1.9%) |

**Table 4. New pipeline's validation results relative to the expressed protein changing variants identified by the previous pipeline. The target variants column shows the number of expressed protein changing variants previously identified. The next columns contain the number of target variants called by each variant caller in the new pipeline. The intersection presents the number of target variants that were called by all variant callers. The values in the column of (variants) requiring inspection were obtained by subtracting the number of target variants with that of the intersection added to the value in the last column. The last column shows the number of target variants that, although called, were filtered by all variant callers. Values marked with an asterisk (*) include one correctly reported MNV (in the respective sample) that corresponds to an SNV in the previous pipeline. The two asterisks (**) concern three variants that, after a second visual inspection were considered false positives and removed from the target variants list in future analysis.**

Using these four metrics – TP, FP, TN, FN – we can calculate sensitivity (true positive rate) and specificity (true negative rate), given by Equations 1 and 2, respectively. Table 2 shows the sensitivity and specificity of each variant caller running on the synthetic dataset. Table 3 shows the sensitivity and specificity of the union of all variant callers and of only Mutect2 and Strelka2 running on the synthetic dataset. This table highlights VarDict's negative impact on specificity caused by an high FP count. Nonetheless, having achieved a sensitivity of 0.991, we considered that the variants called by the union of all variant callers was good. As we are especially interested in maximizing the number of detected variants and considering some aspects of this dataset were not ideal (not covered here), we regard the validation as successful even if VarDict has a lower specificity than desired.

### Previous Results

A major focus in developing the new pipeline was maintaining coherency with the results from the previous pipeline. This would allow continuity in the analysis of the same samples and give the research group confidence in the predicting performance of the new pipeline. In practice, this meant the new pipeline was required to call the same variants as the previous one did.

To confirm whether the requisite above was met, we ran the new pipeline on real, low-mutation CRC patient's tumor-normal paired whole exome sequencing samples that had already been processed by the previous version of the pipeline. Moreover, the predicted variants from the previous pipeline had already undergone visual inspection by the Immunogenomics research group. The visual inspection was done using the Integrative Genomics Viewer (IGV) [47], a genome visualization tool that allows to manually review aligned NGS reads. Briefly, this inspection verifies whether each variant is non-synonymous (codes for different proteins than originally) and that it is being expressed by checking the corresponding RNA sequencing reads. These variants represent the closest possible to a ground truth, and we refer to them as target variants.

Table 4 summarizes the validation results obtained. The variants called by all variant callers (intersection) are considered true and can skip individual inspection. This is identical to the workflow followed by the Immunogenomics group when using the previous pipeline. In the table we see that, across all samples, 91% of the target variants were in the intersection results. The remaining 9% are divided in two categories: requiring inspection and filtered by all. The former are variants that, in a real patient analysis done by the Immunogenomics research group, would have to be individually inspected to assess whether they would be considered true. In these low mutation samples, all the variants are considered even if they are only called by one variant caller. The variants that are not called by the three variant callers (intersection), are visually inspected to discard sequencing artifacts. Furthermore, two of the variants that required inspection (the one in the NIC3 sample and one from NIC7) were correctly reported as MNVs, whereas in the previous pipeline they had been detected as SNVs. Three variants were identified but filtered unanimously by the variant callers (NIC7 sample). The filters were related to either weak evidence, low quality or read bias, or a combination of them. These filters, combined with visual inspection, lead the research group to consider the variants as false in future analysis.

## CONCLUSIONS

By using WDL and Cromwell, we were able to leverage the HPC cluster at LUMC, which was impossible with the previous makefile-based pipeline. This resulted in a steep decrease in execution time, from days to hours (10-14 hours for the samples in Table 4). It would be interesting to collect detailed metrics on execution time, while checking for workflow improvements. However, considering the pipeline is running on an HPC cluster this analysis poses some challenges. Among other aspects, the compute jobs that are submitted to the cluster can end up scheduled in a multitude of heterogeneous nodes and the load on each node will vary depending on the jobs that other users have submitted (compute resources are shared).

Concerning the synthetic validation values for sensitivity in Table 2, we consider the variant callers to have performed well in finding the true variants. However, even though VarDict had the best sensitivity, it called thousands of FP variants, resulting in a subpar value for specificity. For the Immunogenomics research group, this is not problematic because the priority is finding all possible protein changing variants in CRC with low mutation numbers.

Regarding the validation with previous results, we consider that the new pipeline results are in line with the list of variants generated by the previous pipeline (see Table 4). Additionally, if we were to adjust our results to consider the two MNVs as belonging to the intersection, and the three "filtered by all" variants to be excluded from the target variants, the resulting percentage for the intersection would go up to 94.1%. Ideally, more samples should be analyzed.

The integrated pipeline is hosted in the BioWDL GitHub repository, currently in version 4.0.0 [49], where it is maintained and updated with newer tool versions and workflows to keep up with the needs of the Immunogenomics group and other users of the pipeline (our contributions ended in version 1.0.0 [50]). Instructions on the pipeline usage are also available online [31]. Since the GitHub repository is public, anyone can contribute by making pull requests with improvements. The latter also applies to the standalone HLA Binding Prediction module [36] and R Shiny application [18]. All the software developed in this work is open-source, freely available and welcomes contributions.

## ACKNOWLEDGMENTS

We would like to thank Davy Cats, Ruben Vorderman and Dr. Hailiang Mei from the Sequencing Analysis Support Core team at LUMC for sharing their expertise in building pipelines using WDL and Cromwell.

## REFERENCES

1. Uniprot Homepage. `https://www.uniprot.org/`. Last accessed 25 Aug 2019.
2. Andreatta, M., and Nielsen, M. Gapped sequence alignment using artificial neural networks: application to the MHC class I system. *Bioinformatics 32*, 4 (2016), 511–517.
3. Babraham Bioinformatics. FastQC Tool. `https://www.bioinformatics.babraham.ac.uk/projects/fastqc/`. Last accessed 19 Dec 2018.
4. Bauer, D. C., Zadoorian, A., Wilson, L. O. W., Alliance, M. G. H., and Thorne, N. P. Evaluation of computational programs to predict hla genotypes from genomic sequencing data. *Briefings in Bioinformatics 19*, 2 (2018), 179–187.
5. Behjati, S., and Tarpey, P. S. What is next generation sequencing? *Archives of Disease in Childhood - Education and Practice 98*, 6 (2013), 236–238.
6. Benjamin, D., and Sato, T. Notes on Mutect2. `https://github.com/broadinstitute/gatk/blob/master/docs/mutect/mutect.pdf`. Last accessed 26 Aug 2020.
7. Bootstrap Team. Boostrap - The most popular HTML, CSS, and JS library in the world. `https://getbootstrap.com/`. Last accessed 21 Oct 2019.
8. Broad Institute. Base Quality Score Recalibration. `https://gatk.broadinstitute.org/hc/en-us/articles/360035890531-Base-Quality-Score-Recalibration-BQSR`. Last accessed 26 Aug 2020.
9. Broad Institute. Genome Analysis Toolkit. `https://gatk.broadinstitute.org`. Last accessed 26 Aug 2020.
10. Broad Institute. (How to) Call somatic mutations using GATK4 Mutect2. `https://bit.ly/2WcNpwY`. Last accessed 23 Aug 2019.
11. Broad Institute. Picard Tool. `https://broadinstitute.github.io/picard/`. Last accessed 26 Aug 2020.
12. Camacho, C., Coulouris, G., Avagyan, V., Ma, N., Papadopoulos, J., Bealer, K., and Madden, T. L. BLAST+: architecture and applications. *BMC Bioinformatics 10*, 1 (Dec 2009), 421.
13. Docker Hub. `https://hub.docker.com/`. Last accessed 5 Jan 2019.
14. Duan, F., Duitama, J., Seesi, S. A., Ayres, C., Corcelli, S., Pawashe, A., Blanchard, T., McMahon, D., Sidney, J., Sette, A., Baker, B., Mandoiu, I., and Srivastava, P. Genomic and bioinformatic profiling of mutational neo-epitopes reveals new rules to predict anti-cancer immunogenicity. *Journal of Experimental Medicine 211*, 11 (2014), 2231–2248.
15. Ewels, P., Magnusson, M., Lundin, S., and Käller, M. MultiQC: summarize analysis results for multiple tools and samples in a single report. *Bioinformatics 32*, 19 (06 2016), 3047–3048.
16. Finnigan, J., Rubinsteyn, A., Hammerbacher, J., and Bhardwaj, N. Mutation-derived tumor antigens: Novel targets in cancer immunotherapy. *Oncology (Williston Park, N.Y.) 29* (2015).
17. Fritsch, E. F., Rajasagi, M., Ott, P. A., Brusic, V., Hacohen, N., and Wu, C. J. HLA-binding properties of tumor neoepitopes in humans. *Cancer Immunology Research 2*, 6 (2014), 522–529.
18. Frölich, S., Paulo, A., and Ruano, D. R Shiny Module GitHub Repository. `https://github.com/Amfgcp/neoseq_shiny`. Last accessed 12 Sep 2020.
19. Genome Research Consortium. `https://www.ncbi.nlm.nih.gov/grc`. Last accessed 30 Sep 2019.

20. Git Submodules. `https://git-scm.com/book/en/v2/Git-Tools-Submodules`. Last accessed 18 Oct 2019.

21. GNU make. `https://www.gnu.org/software/make/manual/make.html`. Last accessed 24 Oct 2019.

22. Immune Epitope Database and Analysis Resource. MHC I Automated Server Benchmarks. `http://tools.iedb.org/auto_bench/mhci/weekly/`. Last accessed 1 Jan 2019.

23. International Cancer Genome Consortium (ICGC) and The Cancer Genome Atlas (TCGA). ICGC-TCGA DREAM Mutation Calling challenge. `https://www.synapse.org/#!Synapse:syn312572/wiki/58893`. Last accessed 01 Apr 2019.

24. Jurtz, V. I., and Olsen, L. R. *Cancer Bioinformatics*, vol. 1878 of *Methods in Molecular Biology*. Humana Press, New York, NY, 2019, ch. 9, 157–172.

25. Kim, S., Scheffler, K., Halpern, A. L., Bekritsky, M. A., Noh, E., Källberg, M., Chen, X., Kim, Y., Beyter, D., Krusche, P., and Saunders, C. T. Strelka2: fast and accurate calling of germline and somatic variants. *Nat Methods 15*, 8 (08 2018), 591–594.

26. Krekel, H. Pytest Framework. `https://docs.pytest.org/en/latest/`. Last accessed 16 Oct 2019.

27. Lai, Z., Markovets, A., Ahdesmaki, M., Chapman, B., Hofmann, O., McEwen, R., Johnson, J., Dougherty, B., Barrett, C., and Dry, J. Vardict: A novel and versatile variant caller for next-generation sequencing in cancer research. *Nucleic Acids Research 44* (2016), e108.

28. Immunogenomics Group, Pathology Department, LUMC. `https://www.lumc.nl/org/pathologie/research/90708043159185/1709335/`. Last accessed 23 Nov 2020.

29. Li, H. Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. *arXiv: Genomics* (2013).

30. Liu, L., Li, Y., Li, S., Hu, N., He, Y., Pong, R., Lin, D., Lu, L., and Law, M. Comparison of next-generation sequencing systems. *Journal of biomedicine & biotechnology 2012* (2012), 251364–251364.

31. LUMC Sequencing Analysis Support Core. BioWDL Pipeline Usage (Release 1.0.0). `https://biowdl.github.io/germline-DNA/v1.0.0/index.html`. Last accessed 09 Sep 2020.

32. Maleki Vareki, S. High and low mutational burden tumors versus immunologically hot and cold tumors and response to immune checkpoint inhibitors. *Journal for immunotherapy of cancer 6*, 1 (Dec 2018), 157–157.

33. Martin, M. Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet.journal 17*, 1 (2011), 10–12.

34. Ott, P. A., Hu, Z., Keskin, D. B., Shukla, S. A., Sun, J., Bozym, D. J., Zhang, W., Luoma, A., Giobbie-Hurder, A., Peter, L., Chen, C., Olive, O., Carter, T. A., Li, S., Lieb, D. J., Eisenhaure, T., Gjini, E., Stevens, J., Lane, W. J., Javeri, I., Nellaiappan, K., Salazar, A. M., Daley, H., Seaman, M., Buchbinder, E. I., Yoon, C. H., Harden, M., Lennon, N., Gabriel, S., Rodig, S. J., Barouch, D. H., Aster, J. C., Getz, G., Wucherpfennig, K., Neuberg, D., Ritz, J., Lander, E. S., Fritsch, E. F., Hacohen, N., and Wu, C. J. An immunogenic personal neoantigen vaccine for patients with melanoma. *Nature 547* (2017), 217–221.

35. Paulo, A., Frölich, S., and Ruano, D. R Shiny Module Web Deployment. `https://neoseq.shinyapps.io/shiny/`. Last accessed 12 Sep 2020.

36. Paulo, A., and Ruano, D. Binding Prediction Module GitHub Repository. `https://github.com/Amfgcp/NeoSeq_WDL/`. Last accessed 09 Sep 2020.

37. Quail, M. A., Smith, M., Coupland, P., Otto, T. D., Harris, S. R., Connor, T. R., Bertoni, A., Swerdlow, H. P., and Gu, Y. A tale of three next generation sequencing platforms: comparison of ion torrent, pacific biosciences and illumina miseq sequencers. *BMC genomics 13* (Jul 2012), 341–341.

38. R Studio. R Shiny. `https://shiny.rstudio.com/`. Last accessed 21 Oct 2019.

39. Red Hat. Quay Container Images. `https://quay.io`. Last accessed 21 Oct 2019.

40. Reynisson, B., Alvarez, B., Paul, S., Peters, B., and Nielsen, M. 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 (05 2020), W449–W454.

41. Rubinsteyn, A. Mhctools python package. `https://pypi.org/project/mhctools/`. Last accessed 02 Sep 2019.

42. Rubinsteyn, A., Kodysh, J., and Aksoy, B. A. hammerlab/isovar: Version 0.7.0. `https://doi.org/10.5281/zenodo.821224`, 2017.

43. Rubynstein, A., Paulo, A., and Ruano, D. Modified mhctools (GitHub Fork webpage). `https://github.com/Amfgcp/mhctools`. Last accessed 09 Sep 2020.

44. Sahin, U., Derhovanessian, E., Miller, M., Kloke, B.-P., Simon, P., Löwer, M., Bukur, V., Tadmor, A. D., Luxemburger, U., Schrörs, B., Omokoko, T., Vormehr, M., Albrecht, C., Paruzynski, A., Kuhn, A. N., Buck, J., Heesch, S., Schreeb, K. H., Müller, F., Ortseifer, I., Vogler, I., Godehardt, E., Attig, S., Rae, R., Breitkreuz, A., Tolliver, C., Suchan, M., Martic, G., Hohberger, A., Sorn, P., Diekmann, J., Ciesla, J., Waksmann, O., Brück,

A.-K., Witt, M., Zillgen, M., Rothermel, A., Kasemann, B., Langer, D., Bolte, S., Diken, M., Kreiter, S., Nemecek, R., Gebhardt, C., Grabbe, S., Höller, C., Utikal, J., Huber, C., Loquai, C., and Türeci, Ö. Personalized RNA mutanome vaccines mobilize poly-specific therapeutic immunity against cancer. *Nature 547* (2017), 222–226.

45. Sylabs. Singularity Container Platform. `https://www.sylabs.io`. Last accessed 5 Jan 2019.
46. Szolek, A., Schubert, B., Mohr, C., Sturm, M., Feldhahn, M., and Kohlbacher, O. Optitype: precision hla typing from next-generation sequencing data. *Bioinformatics 30*, 23 (2014), 3310–3316.
47. Thorvaldsdóttir, H., Robinson, J. T., and Mesirov, J. P. Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration. *Briefings in Bioinformatics 14*, 2 (04 2012), 178–192.
48. Travis Continuous Integration Platform. `https://travis-ci.com/`. Last accessed 5 Jan 2019.
49. Vorderman, R., Cats, D., van 't Hof, P., Agaser, C., Paulo, A., and Mei, L. Biowdl pipeline github repository release 4.0.0. `https://github.com/biowdl/germline-DNA/tree/v4.0.0`, Aug 2020.
50. Vorderman, R., van 't Hof, P., Cats, D., Paulo, A., and Mei, L. Biowdl pipeline github repository release 1.0.0. `https://github.com/biowdl/germline-DNA/tree/v1.0.0`, Sep 2019.
51. Voss, K., Gentry, J., and Van der Auwera, G. Full-stack genomics pipelining with GATK4 + WDL + Cromwell. *F1000Research 6(ISCB Comm J):1379 (poster)* (2017).
52. Wilm, A., Aw, P. P. K., Bertrand, D., Yeo, G. H. T., Ong, S. H., Wong, C. H., Khor, C. C., Petric, R., Hibberd, M. L., and Nagarajan, N. Lofreq: a sequence-quality aware, ultra-sensitive variant caller for uncovering cell-population heterogeneity from high-throughput sequencing datasets. *Nucleic Acids Res 40*, 22 (2012), 11189–11201.
53. Xu, C. A review of somatic single nucleotide variant calling algorithms for next-generation sequencing data. *Comput Struct Biotechnol J 16* (2018), 15–24.
54. Xu, C., Nezami Ranjbar, M. R., Wu, Z., DiCarlo, J., and Wang, Y. Detecting very low allele fraction variants using targeted dna sequencing and a novel molecular barcode-aware variant caller. *BMC Genomics 18*, 1 (Jan 2017), 5.
55. Xu, H., DiCarlo, J., Satya, R. V., Peng, Q., and Wang, Y. Comparison of somatic mutation calling methods in amplicon and whole exome sequence data. *BMC Genomics 15*, 1 (2014), 244–253.

## SUPPLEMENTARY MATERIAL

### Pipeline WDL Workflows Diagrams

Using Unified Modeling Language (UML) sequence diagrams, we show the process flow and interactions between the WDL workflows of the integrated pipeline. We made two adaptations to the diagrams that are outside of the current UML specification (version 2.5.1, 2017):

1. We make use of composition arrows (distinguished by the filled in diamond on one end). They convey a multiple simultaneous call to the same workflow or task, serving as an abstraction to the occurrence of input division into smaller pieces. Usually, this means scattering the genome into smaller regions so that each job (run in parallel) takes less time to complete.
2. The parallel combined fragment in Figure S3 is divided vertically instead of horizontally to save vertical space.

In the following diagrams (Figures S1-S3), lifelines (horizontal rectangles on the top with text and a dashed line descending from them) correspond to WDL workflows that call tasks throughout their execution specifications (vertical grey rectangles). Some data transformation and optional tasks were omitted to achieve a clearer diagram. With the same goal, arrows returning from workflows show only the most relevant returned outputs, and some workflow and task names were slightly changed when compared to the source code.

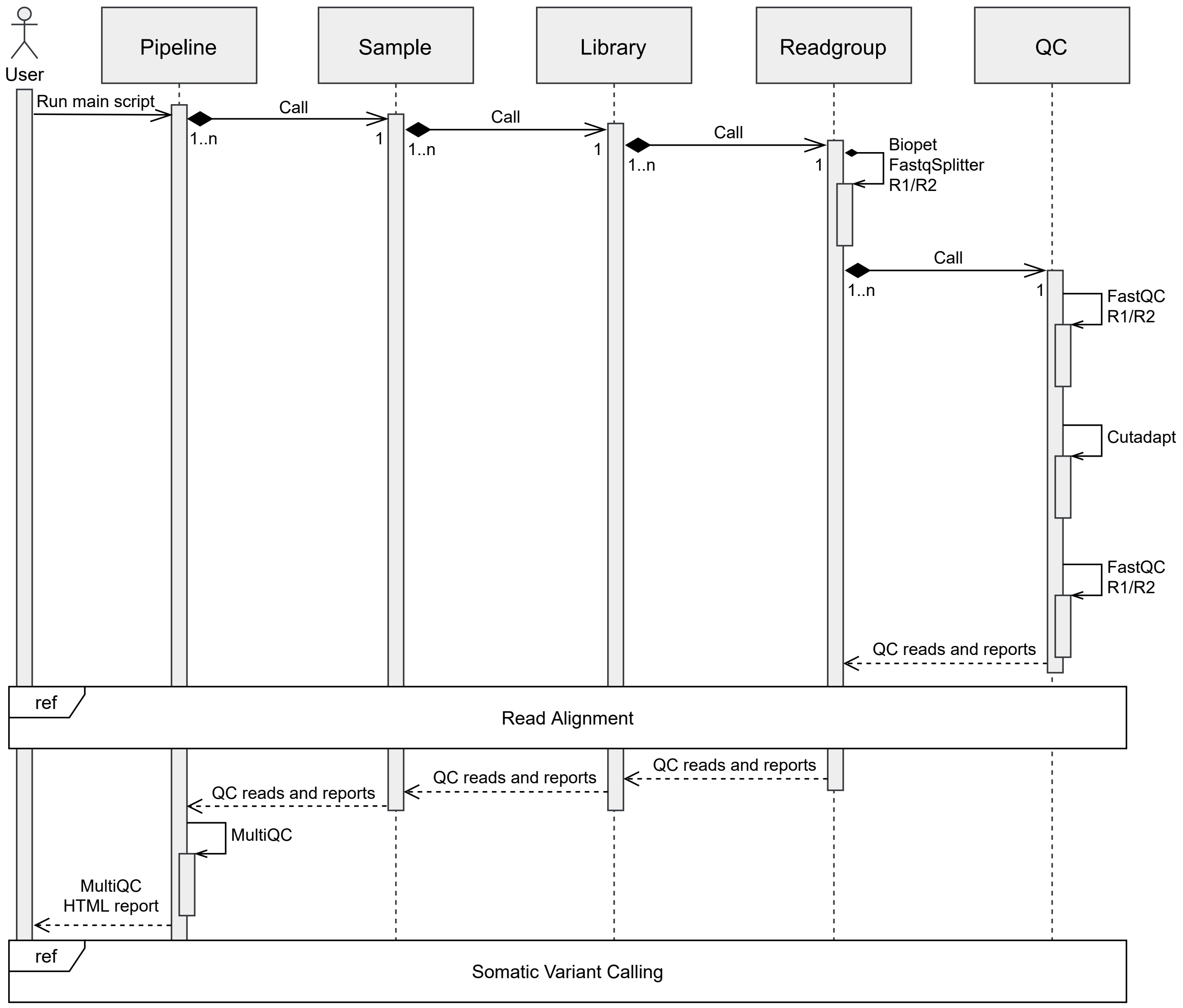


**Figure S1. QC and Adapter Trimming WDL workflow.**

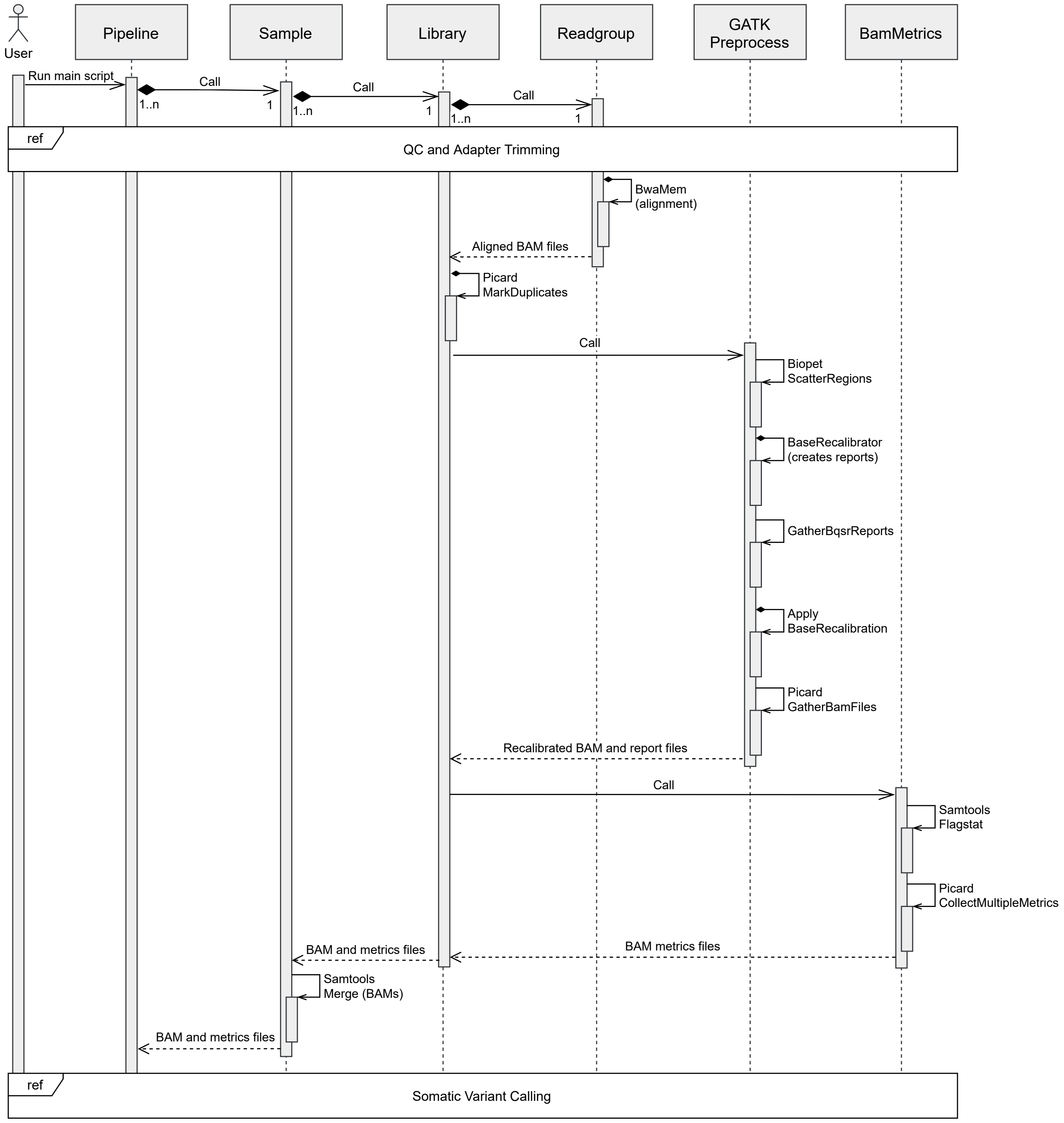


**Figure S2. Read Mapping WDL workflow.**

## Data Visualization

There are two ways that data visualization is addressed: 1) quality control plots generated from pipeline collected metrics regarding the sequencing reads. 2) Somatic variants amount and type plots, generated by the R Shiny module. The data is obtained from processing the Somatic Variant Calling module output files.

### *Pipeline Generated Plots*

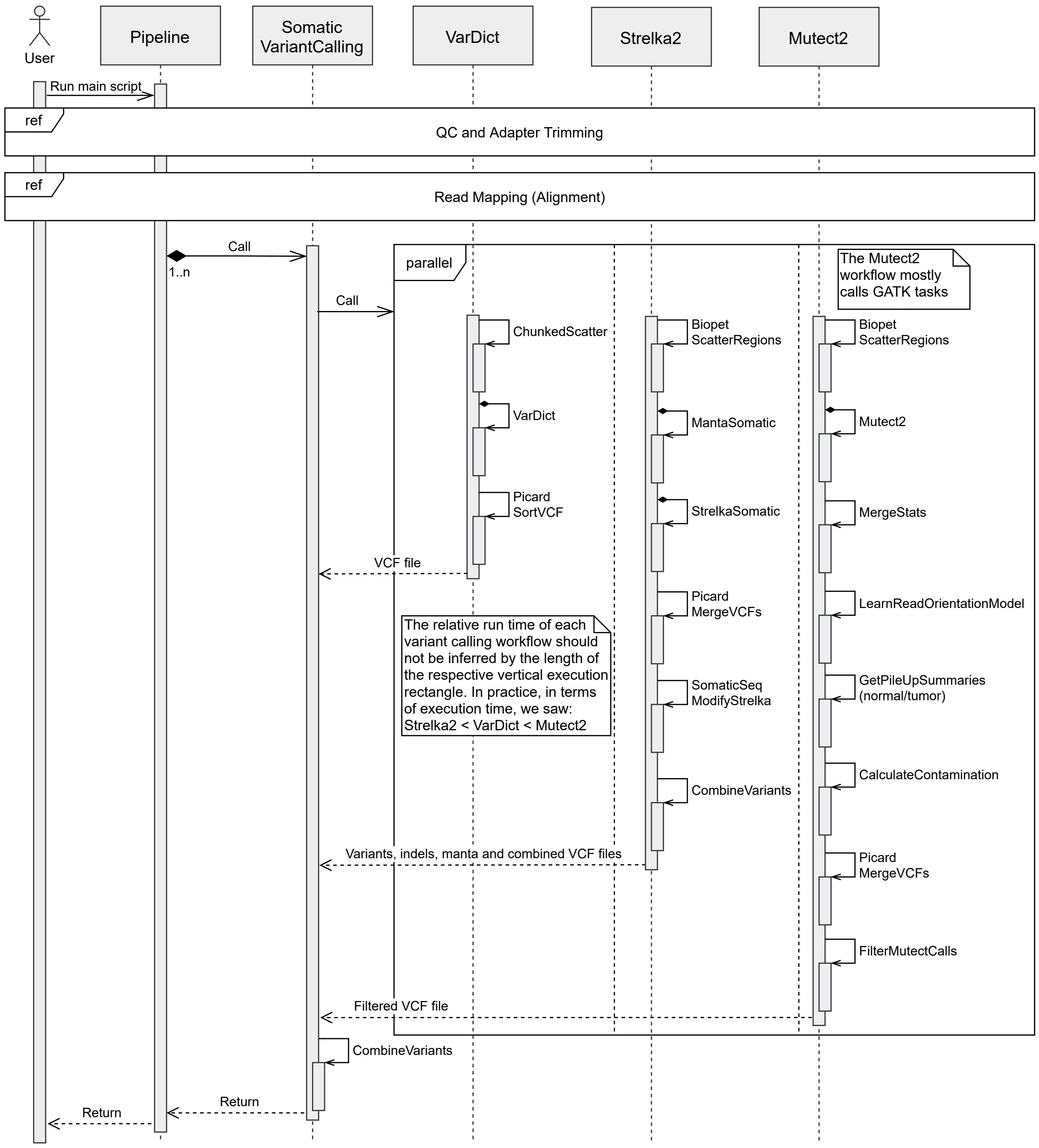


**Figure S3. Somatic Variant Calling WDL workflow.**

Figure S4 shows the "Mean Quality Scores" plot in MultiQC, generated from FastQC data. This figure showcases MultiQC's sample highlighting (lines in blue). MultiQC also allows filtering out samples from the plots, zooming in on any given part of the plots, exporting the plots to different formats and saving the applied viewing options. In Figure S4, we see that there is a left side bar with links to not only other FastQC plots (bottom), but also to plots presenting data from other tools used in the pipeline – Picard, SAMtools and GATK – as MultiQC automatically gathers all reports found in the specified folder (and its subfolders).

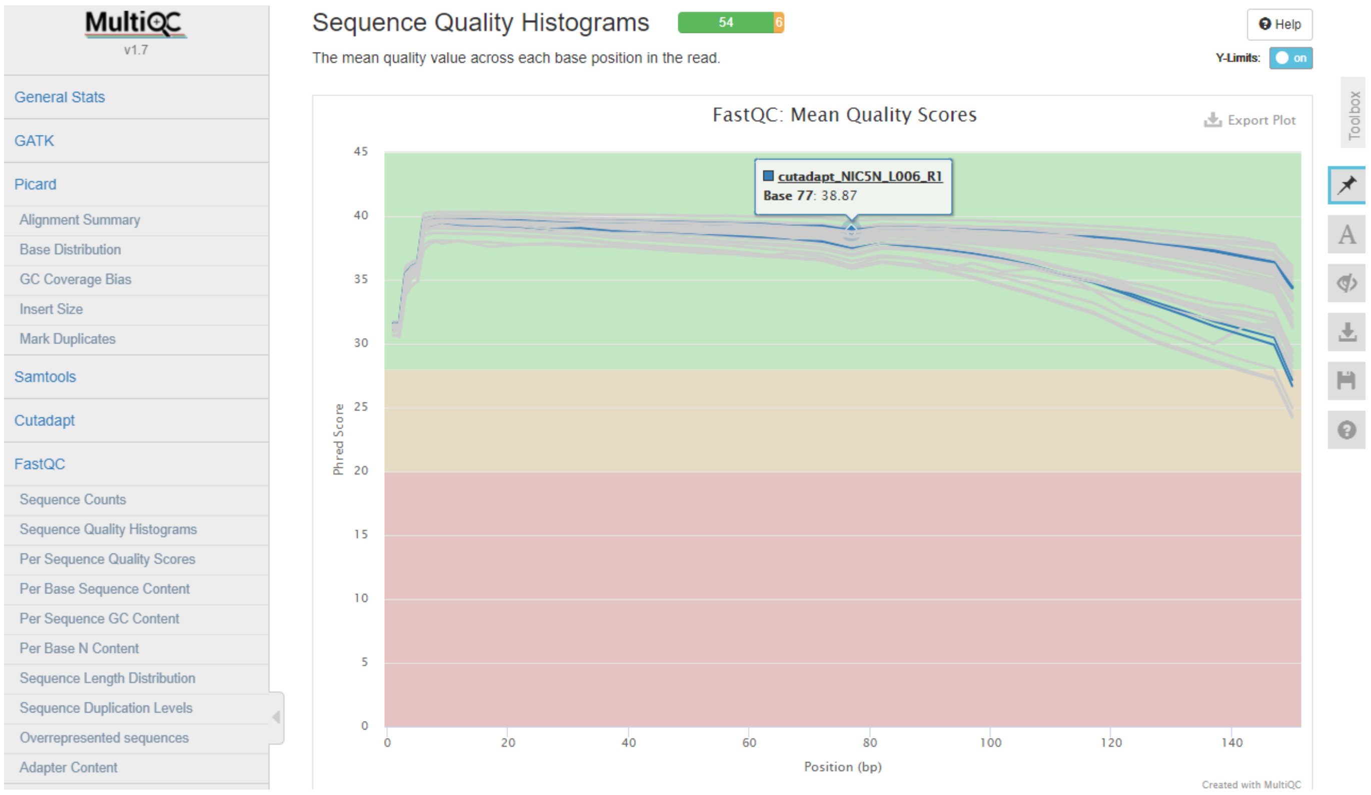
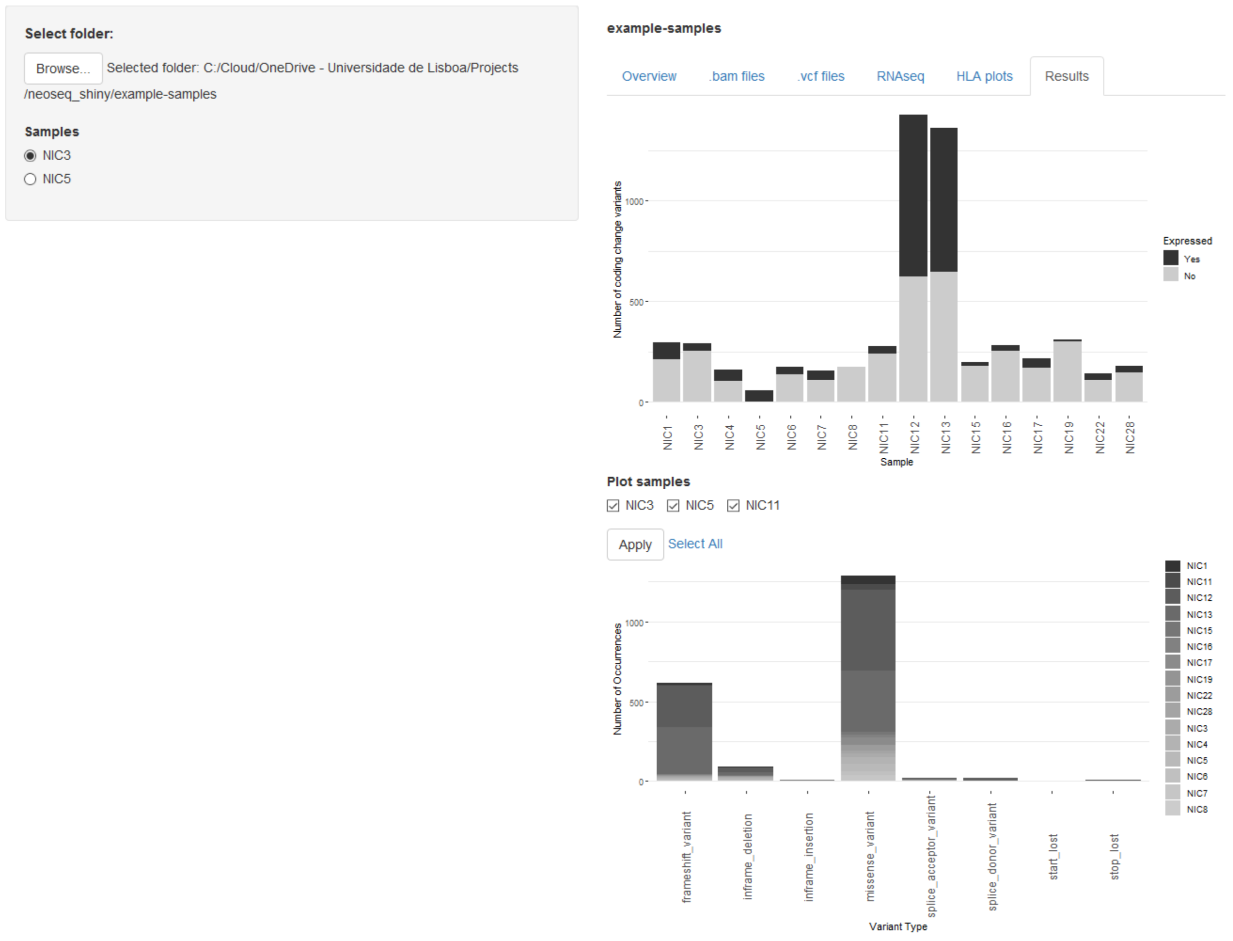


**Figure S4. MultiQC's "FastQC: Mean Quality Scores" plot with samples highlighted in blue.**

*R Shiny Module*

An overview is shown in Figure S5: on the top left, the user can browse directories and is expected to select one with a specific structure. Then, the data scraping script locates and processes the files containing the variants and the resulting peptides. One plot shows the number of protein changing variants per sample and whether they are expressed (top right). The other plot shows the type of variants (bottom right).


**Select folder:**

Browse... Selected folder: C:/Cloud/OneDrive - Universidade de Lisboa/Projects /neoseq_shiny/example-samples

**Samples**

[✓ NIC3] NIC5

**example-samples**

**Plot samples**

[✓ NIC3] [✓ NIC5] [✓ NIC11]

**Figure S5. R Shiny module overview. On the left, the user can browse folders. On the right, the plots generated with the data from the chosen folder are shown.**