

IGV-*plus*: A Java Software for the Analysis and Visualization of Next-Generation Sequencing Data

Antonio Agliata, Marco De Martino, Maria Brigida Ferraro,
and Mario Rosario Guarracino

Abstract In this work we describe IGV-*plus*, a software for next-generation sequencing (NGS) data analysis and visualization. It integrates de facto standard tools for the discovery of genetic mutations in genomic-wide association studies. We describe the software specification that led to the development of IGV-*plus*. Finally, we show how we integrate a single-nucleotide polymorphism (SNP) calling software of the genome analysis toolkit (GATK) in the genome browser integrative genomics viewer (IGV), in order to create a centralized platform, as a possible one-stop shop for biologists dealing with NGS data.

1 Introduction

High-throughput technologies for genome sequencing, namely next-generation sequencing (NGS) platforms [1], produce large amounts of biological data. Such data need extensive preprocessing before they can be used for genome-wide association studies, in which genetic variants are detected and correlated to specific traits. The availability of integrated software tools for the analysis, comparison, view, and annotation of such data becomes a discriminating factor for the success of a biological analysis [2].

The idea of NGS technology is similar to capillary electrophoresis (CE)-based Sanger sequencing. The bases of a small fragment of DNA are sequentially identified from signals emitted as each fragment is resynthesized from a DNA template strand. NGS extends this process across millions of reactions through a massive parallelization. It provides an enormous number of reads, which permits the sequencing of entire genomes at a fraction of the costs for Sanger technology. The main steps in NGS data preprocessing consist in computing the quality of base calls,

A. Agliata (✉) • M. De Martino • M.R. Guarracino
High Performance Computing and Networking Institute, National Research Council, Naples, Italy
e-mail: agliataantonio@gmail.com

M.B. Ferraro
Department of Statistical Sciences, Sapienza University of Rome, Rome, Italy

aligning the short-reads to a reference genome (when available), evaluating quality of such alignment for each short-read, and recalibrating the base calls quality in the context of the aligned reads. After this phase, the discovery of genomic variants is executed, and scientists visualize and analyze results using genome browsers. This visual analysis shows the nucleotide context in which variants occur and other factors of biological interest, such as splicing sites, promoters, terminators, noncoding RNAs, and repeated sequences. Although many software are available for each of such tasks and standards are emerging, some efforts are needed to use the output of one software as the input of the next. To our knowledge, no single software integrates all those steps, thus enabling the users to control the complete process underlying their analysis.

In this work we describe *IGV-plus*, an open-source software that integrates existing solutions for data preprocessing, genetic variant calling, and genome browsing, providing a single tool for each and all the steps needed in NGS data analysis. *IGV-plus* is extensible to plug in other existing tools, given they meet some general requirements. Our analysis of existing software starts from preprocessing and variant discovery software that are noncommercial and open source, with support for input files in BAM and SAM formats [3] and output in VCF format [4]. Furthermore, we only take into consideration those supporting pooled analysis of individuals. Then we analyze existing genome browsers to upload, view, and explore the alignment of the datasets to a reference genome [5]. There exist tens of genome browsers, as a simple web search will reveal. Nevertheless, only a few are widely adopted by the scientific community, usable across different operating systems, freely available and downloadable together with their source code, easy to use and with intuitive graphical user interface (GUI), executable on standard desktop computers, and supported by a developers' community. As a result of our analysis, we decided to integrate the preprocessing and the variant calling software GATK [6] with the genome browser integrative genomics viewer (IGV) [7].

The paper is organized as follows. In Sect. 2 we report a software specification based on the analysis of user needs (Sect. 2.1). In particular, a simple use case diagram is showed and the execution of the main tasks is described. In Sects. 2.2 and 2.3 we analyze tools for single-nucleotide polymorphism (SNP) calling and the existing software solution to browse genomes. In Sect. 3 we provide the implementation details, by analyzing the changes in Java class and the additional packages required for the integration. Finally, in Sect. 4 conclusions are discussed and future work is addressed.

2 Software Specification

Analysis tools in the preprocessing phase are needed to detect positions of mutations among billions of possible ones. Using only a genome browser software would mean manually searching and viewing all data position, which is a daunting and prone-to-error task. Genome browser is useful in the next phase to view the position filtered by preprocessing and analyze them with useful annotation information.

So the goal of IGV-plus is creating a centralized platform for biologists dealing with NGS analysis by the integration of analysis tool into genome browser.

In this section we describe the biologists' requirements, the choice of the genome viewer, and the analysis tool to obtain an integrated software.

2.1 User Requirements

Not only do biologists require the list of candidate mutations, but they also need to visually analyze these positions. After the identification of possible causative mutations, they start the biological validation of those that might be of interest to confirm their hypothesis. This is usually done with Sanger sequencing [8] or other techniques, which are expensive in terms of time, resources, and expertise. For this reason, they require to exclude variants that are obviously unrelated with the scope of their research, and artifacts introduced in the amplification process before the sequencing step, to avoid biological validations that would be unnecessary.

From these requirements, a simple use case diagram has been drawn (Fig. 1), where the actor is represented by the software user and it is assumed to be of only one type. In addition to the standard functions performed by a genome viewer, the user can conduct a new analysis and/or evaluate the results of an existing one.

In the *Execute Analysis* task, the user goal is to perform an analysis of the data. The preconditions are that the user is running the IGV-plus and executing the "RunTool" task. The success is obtained when the input form is correctly filled in, and the analysis is started. An exception is raised in case input forms are incorrectly filled in. In case of success, it generates a VCF output file. In the *View Results* task, the user goal is to evaluate the results of an analysis carried out earlier or by others. The preconditions require a VCF file and to execute the "Run Tool" task. The success is obtained when the input form is properly filled in, and the input

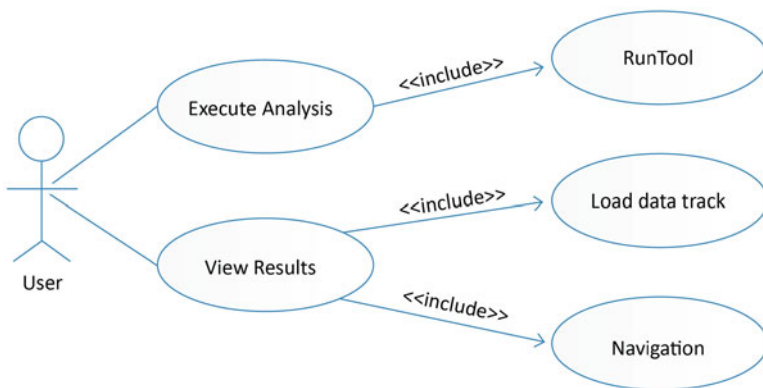


Fig. 1 Use cases diagram

Table 1 Analysis tools: a comparison

Features	GATK	FreeBayes	SNVer	CRISP
Language	Java	C++	Java	Python
Multi-platform	Yes	No	Yes	Yes
Noncommercial	Yes	Yes	Yes	Yes
File format	BAM, BED, FASTA VCF, tab-delim	BAM, FASTA VCF	BAM, BED, FASTA VCF, tab-delim	BAM, FASTA VCF
Supported	Update 2013	Update 2013	Update 2012	Update 2012

file is correctly loaded. An exception is raised in case input forms are incorrectly filled in. The use case describes the evaluation of analysis results that are available in the VCF format. In case of success, the user will be able to browse candidate mutations by including the “Load Data Track” and “Navigation” tasks.

2.2 Preprocessing and Variant Calling Software

We compare four variant calling software: GATK, FreeBayes [9], SNVer [10], and CRISP [11]. The features of these software are described in Table 1.

The choice of variant calling software to integrate falls on GATK and depends on different reasons. First, GATK provides more accurate results of sensitivity and specificity on pooled NGS data [12]. In addition, as we will detail in the following, it is not only a simple variant calling software but also a real analysis framework that provides functions also needed in the preprocessing phase of the data. Finally, it is possible to generate BAM files for other software.

GATK is a software developed at the *Broad Institute* to analyze NGS data. GATK is not only used to perform genomic variant calling, but also integrate analysis to evaluate variants (see Fig. 2). It is a structured framework designed to facilitate the development of efficient and robust analysis tools for NGS data, and it is suitable to be used in projects of any size [6]. GATK is developed in Java and follows the *MapReduce* paradigm that allows the parallelization and distribution of processing by splitting the computation into two steps: *Map* splits the initial problem into independent subproblems and *Reduce* solves the subproblems and combines the partial solutions to get the overall solution to the main problem. The use case describes the evaluation of an analysis results, available in the VCF format.

The currently available version of GATK, 2.7-2, allows us to use two different walkers for this purpose: *Unified Genotyper* and *Haplotype Caller*. Since *Haplotype Caller* does not yet support pooled data, we use the first one.

Searching SNPs in pooled data requires methods capable to distinguish mutations from sequencing errors, mainly with medium-low coverages, and amplification artifacts. With an error rate equal to 1%, a mutation can be easily mistaken for sequencing error.

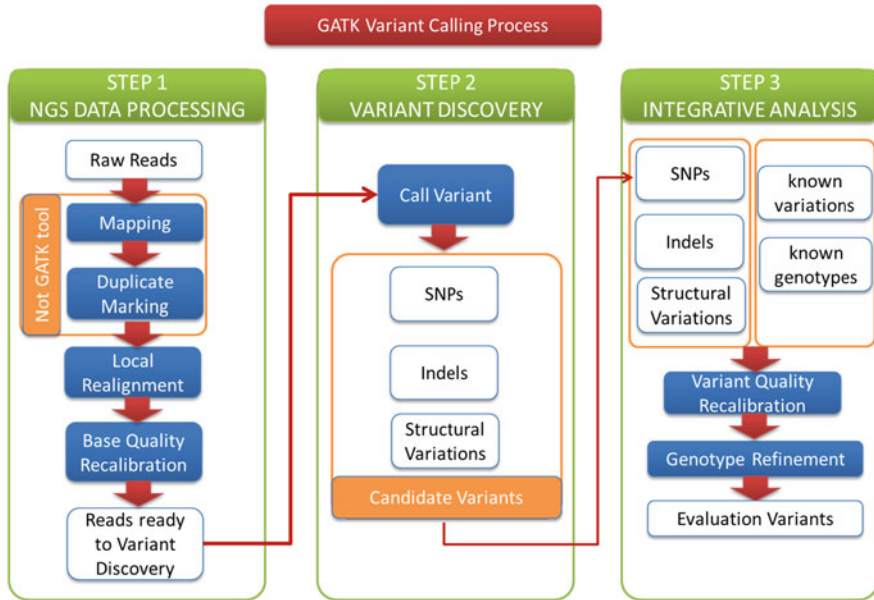


Fig. 2 Calling variants with GATK

The use of the pooling with a good sequencing quality can be very useful to make a proper distinction. Assuming that the reads of each pool are aligned to the reference genome, for each position, we consider the reads of all pools that cover that position. Since GATK it is written in Java, it is executed from the command line.

2.3 Genome Browser

With respect to the choice of the genome viewer software, it derives from the analysis summarized in Table 2.

Although, as shown in Table 2, the software under analysis have characteristics very similar to each other, we choose IGV for the integration, especially for the support it offers for remote control. This feature is very important in an area where large datasets produced by NGS are complex to manage. This means that, when started, IGV runs a web server daemon accepting remote requests. This function can be used to send commands to IGV and to receive results. This choice is also supported by an idea of future development to be implemented, that is, to realize an NGS analysis platform as a web service. The latter facilitates the management of the data, avoiding its duplication among the groups of biologists and of analysts and all those people involved in the project.

Table 2 Visualization software: a comparison

Features	IGV	Savant [13]	Artemis [14]
Open source	Yes	Yes	Yes
Language	Java	Java	Java
Platform	All	All	All
Technology	Illumina, 454, Sanger, ChIP-Seq, RNA-Seq	Illumina, 454, Sanger	Illumina, 454, Sanger
File format	BAM, SAM, GOBY, BED, GFF, GTF, PSL, CN, GCT, FASTA, . . . , . . .	SAM, BAM, FASTA, WIG, GFF, BED, tab-delimited	BAM, VCF, BCF, FASTA, tab-delimited
Remote control	HTTP, HTTPS, FTP	No	No
Supported	Yes	Yes	Yes

Other aspects for which IGV has been chosen are the high support to various file formats and, last but not least, a more complete documentation from the implementation point of view.

IGV is a high-performance viewer for genomic data, capable of handling large heterogeneous datasets while providing a simple and intuitive interface for navigation in the genome [7]. It is written in Java, so it is multi-platform. It supports dataset loading from both local and remote storage. Furthermore, it is available for free under the GNU LGPL license and the required hardware resources allow its installation even on desktop computers. IGV also enables interaction with the data through different levels of detail managed by zoom, which goes from the entire genome to the single base/nucleotide, using an approach similar to that used by *Google Maps*. It uses preprocessed images of genomic data, representing different resolutions, leaving the display at runtime with a better resolution only for the required parts. This approach is called *data tiling* (see Fig. 3).

The software architecture of IGV is divided into three conceptual levels (see Fig. 4). The *application layer* deals with the management of IGV main window and the interaction of the user with the interface elements. IGV displays data in rows that are called *tracks*. They appear in the data panel that handles the layout and the rendering, as well as events related to shared actions, such as zooming. It delegates to object tracks that may refer to object renderers, instanced at runtime, events related to navigation, loading of features, and track design. The *data layer* reads and handles different file formats and makes the data available to the application layer for displaying and on-demand indexing. This allows the optimization of computer resources at runtime. It also creates the data caching to improve efficiency, when requesting a genomic region that is already displayed in the same session. Finally, the *stream layer* manages all the protocols that IGV uses for local and remote access to files and for uploading of the reference genome.

As for all the genome viewers, IGV is mainly used to check the alignment of the analyzed data to the reference genome. Since in the alignment process artifacts



Fig. 3 Integrative genome viewer GUI

are introduced by the sequencing platform, SNPs must be detected from errors. The misalignment, in practice, is suitably highlighted, at certain zoom levels. IGV also offers the ability to display different characteristics of the aligned reads such as mapping quality, the alleles frequency, and many other features used by biologists to validate mutations. Finally, it supports about 30 different input file formats.

3 Implementation Details

In this section we detail the integration of *UnifiedGenotyper*, a specific GATK walker, with IGV. There are two possibilities: integrating the entire GATK code, which is open source, or allowing IGV to execute an instance of GATK. The adopted solution enables the genome viewer to run instances of GATK, whereas the code integration is not convenient in terms of usability and code complexity. Furthermore, this option would make the GUI strictly linked to one version of GATK.

The implementation can be divided into two phases. The first one is reengineering IGV to integrate the variant caller by means of the identification of the package and the integration of the software. The second one consists of developing a GUI that allows the use of GATK.

3.1 IGV Reengineering

The reengineering of IGV starts with its structural analysis. We have identified the packages and classes of interest related to the management of the user interface from the class diagram of IGV (Fig. 4), provided in the documentation. Hence, most of

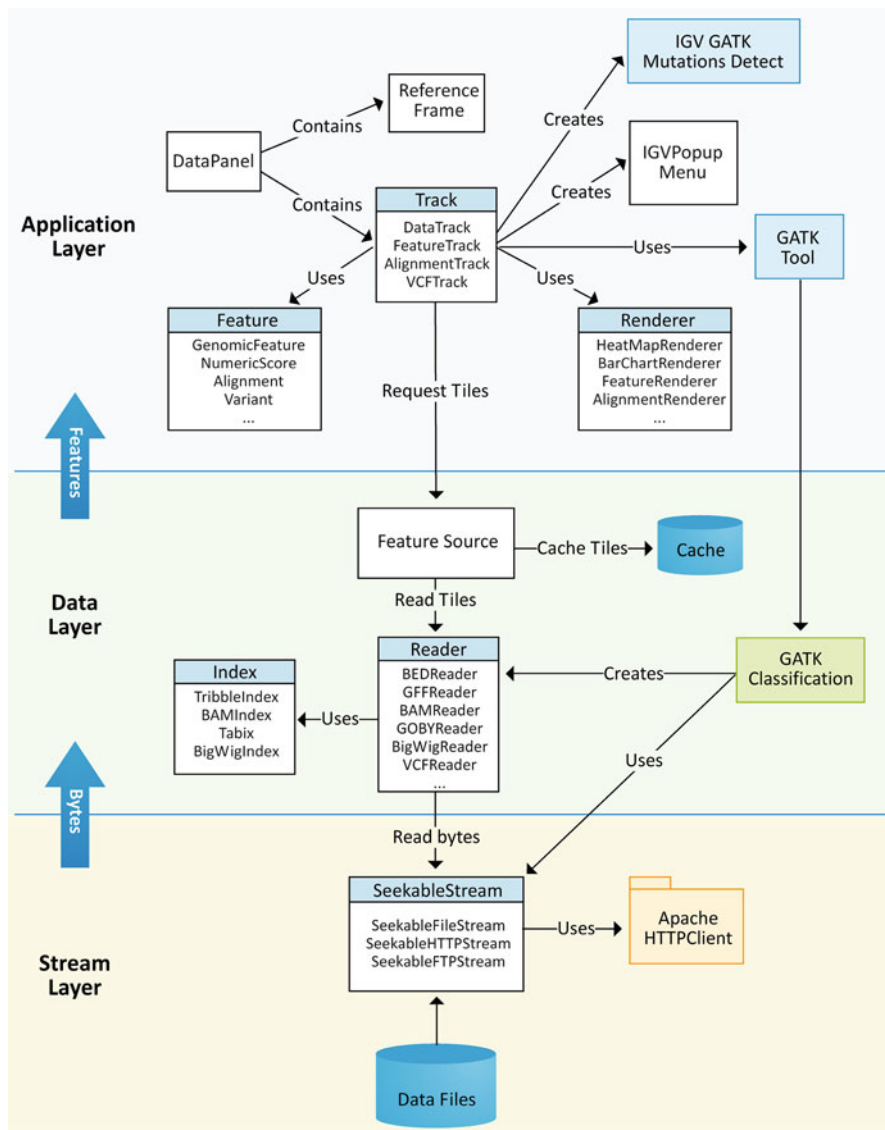


Fig. 4 IGV class diagram



Fig. 5 Menu bar of IGV

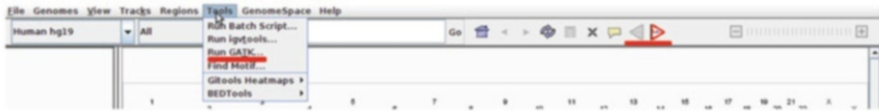


Fig. 6 New menu bar of IGV

the necessary changes of IGV are basically related to interface and, in particular, the command bar and some other menu items (Fig. 5).

The package *org.broad.igv.ui* consists of all the classes dealing with the management of the graphical interface of IGV, the display preferences, and all the involved aspects and graphical events. The goal is to integrate a menu item to run the GATK GUI and to add two buttons *forward* and *back* for browsing the candidate mutations arising from the analysis.

The *IGVMenuBar.java* class deals with the management of the GUI main frame, bringing together items of the main menu and the command bar, in which we want to add the menu item *Tools/Run GATK*. It extends the class *JComponent* that implements the interfaces for the management of mouse events and keyboard for the items in the menu. The *IGVCommandBar.java* deals with the management of the IGV command bar, handling the events of mouse and keyboard inherent to the buttons that appear on the toolbar. Such buttons allow to choose the reference genome, to click the chromosome, to zoom, and other controls to which we want to add the navigation buttons for the candidate mutations.

The edit of the above described classes leads to the addition of new features in IGV (see Fig. 6)

3.2 GATK Graphical User Interface

The capabilities of GATK are structured in two levels: *traversals* and *walkers*. The *traversals layer* is composed of the management modules for common functions. These deal with partitioning and preparation of the data for analysis to be passed to the walkers. The partition of any amount of data is a fundamental problem for the scalability, memory consumption and parallelization task. GATK splits the data into fragments called *shards* whose size is defined by GATK engine and depends on the characteristics of the input BAM file. The *walkers layer* is composed of the management modules of specific functions receiving data from traversals and it

applies the MapReduce paradigm. For example, the function *SNP calling*, used in this work, operates in a Map level as it performs independent processing for each position of the genome. Furthermore, walkers can also run on specific ranges of input files, allowing users to perform analysis only on regions of interest. Each thread performs independently one *MapReduce* call on a single instance of the walker and GATK merges the results of the step *Reduce* of each thread in sequential order, returning the overall step *Reduce*.

The walkers are the core of the functions offered by GATK, making it an essential software to conduct analysis of NGS data even in the case where different tools for the variant calling are used. In the following, we show how to implement a GUI for the *variant calling* walker. This interface provides a GATL function within IGV.

The GATK GUI requires the implementation of new packages for GUI, the management of VCF file format, and the GATK execution command, which are implemented in *org.broad.igv.gatk.ui (Gatk.java)*, *org.broad.igv.gatk.vcf (ReadVcfFile.java, VcfBean.java)*, and *org.broad.igv.gatk.run (RunGatk.java)*, respectively.

Gatk.java deals with all the aspects related to the implementation of the GUI, with the construction of the GATK execution string command and with the management of all parameters that it is possible to set. It provides a robust control on entered inputs, properly reporting the wrong and/or missing inputs. The GUI also integrates tooltips menu for each parameter and each option that contains information on the use and meaning of each of them, taken from the manual. *ReadVcfFile.java* contains the implementation of a parser for VCF files. It reads the mutations reported in the VCF file and its properties, setting proper variables. *VcfBean.java* implements a Java bean for the input fields, by means of *get* and *set* for their management. *RunGatk.java* deals with the GATK command that receives input from the GUI, redirecting the standard output from GUI.

The GUI is organized in different areas. As shown in Fig. 7, A area contains forms to select GATK jar for the execution walker, input BAM file to analyze, input humane genome FASTA file to run GATK analysis, input VCF file of known SNPs, and output VCF file path for called variants. If properly inserted, text box is highlighted in green, otherwise in red. In B area it is possible to set GATK command options like memory usage, number of threads to use in the execution, and input interval to analyze. In C area we can set basic parameters related to the *UnifiedGenotyper* walker, to change the default values. In D area we can set advanced *UnifiedGenotyper* parameters, activated by Advanced checkbox. In E and F areas there are two buttons to run GATK command with selected parameters and to view results in IGV. Finally, in G area, we redirect GATK command standard output, if the execution is successful, and standard errors, otherwise.

To better explain GATK GUI we consider the following execution example. If we want to run a complete NGS analysis we can fill in input forms with GATK jar, input file BAM, human genome FASTA file, and VCF output file path. Then, we select proper parameters to run GATK jar (area B) and analysis (areas C, D). When selecting the Execute Analysis button all parameters are collected for variant calling and analysis starts if errors are not signaled. At the end of execution, we can evaluate

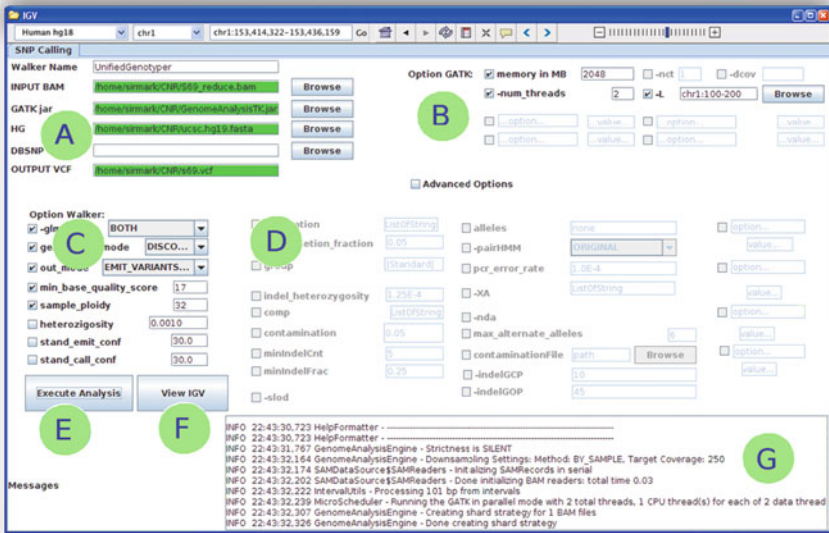


Fig. 7 GATK GUI

GATK execution (area G) and navigate in variants found by clicking the View IGV button that loads input bam file automatically. So we can use the new buttons in IGV GUI to navigate through the called variants.

4 Conclusion and Future Work

In genomic analysis it is necessary to validate the positions of candidate mutations. To reduce the cost of validation, it is useful to view these positions in a software viewer, in order to evaluate also the biological context.

In this work we propose IGV-plus, a software for providing biologists with a simple and centralized platform to make an independent genomic analysis.

It is of great interest for biologists to annotate the discovered mutations in order to create an appropriate report of the visual inspection. Furthermore, in the near future, we will integrate other analysis software and mutation reports, and we will provide them as web services. Finally, we will integrate a database to handle data and associate it to patients, and for storing information about them, as in a laboratory information and management systems (LIMS).

References

1. Nielsen, R., Paul, J.S., Albrechtsen, A., Song, Y.S.: Genotype and SNP calling from next-generation sequencing data. *Nat. Rev. Genet.* **12**, 443–451 (2011)
2. Illumina, Inc.: An introduction to next-generation sequencing technology (2013). <http://www.illumina.com/>
3. The SAM/BAM Format Specification Working Group: Sequence alignment/map format specification. <https://github.com/samtools/hts-specs> (2014)
4. The Variant Call Format (VCF) Version 4.1 Specification. <https://github.com/samtools/hts-specs> (2013)
5. Robinson, J.T., Thorvaldsdottir, H., Winckler, W., Guttman, M., Lander, E.S., Getz, G., Mesirov, J.P.: Integrative genomics viewer. *Nat. Biotechnol.* **29**, 24–26 (2011)
6. McKenna, A., Hanna, M., Banks, E., Sivachenko, A., Cibulskis, K., Kernysky, A., Garimella, K., Altshuler, D., Gabriel, S., Daly, M., DePristo, M.A.: The genome analysis toolkit: a mapreduce framework for analyzing next-generation DNA sequencing data. *Genome Res.* **20**, 1297–303 (2010)
7. Thorvaldsdottir, H., Robinson, J.T., Mesirov, J.P.: Integrative genomics viewer (IGV): high-performance genomics data visualization and exploration. *Brief. Bioinform.* **14**, 178–192 (2012)
8. Sanger, F., Coulson, A.R.: A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. *J. Mol. Biol.* **94**, 441–448 (1975)
9. Garrison, E., Marth, G.: Haplotype-based variant detection from short-read sequencing (2012). <http://arxiv.org/abs/1207.3907>
10. Wei, Z., Wang, W., Hu, P., Lyon, G.J., Hakonarson, H.: SNVer: a statistical tool for variant calling in analysis of pooled or individual next-generation sequencing data. *Nucl. Acids Res.* **39** (2011). doi:10.1093/nar/gkr599
11. Bansal, V.: A statistical method for the detection of variants from next-generation resequencing of DNA pools. *Bioinformatics* **26**, 318–324 (2010)
12. Ferraro, M.B., Guarracino, M.R.: Prediction of single-nucleotide polymorphisms causative of rare diseases. In: *Computational Intelligence Methods for Bioinformatics and Biostatistics*, pp. 213–224. Springer, Berlin (2014)
13. Fiume, M., Williams, V., Brook, A., Brudno, M.: Savant: genome browser for high-throughput sequencing data. *Bioinformatics* **26**, 1938–1944 (2010)
14. Carver, T., Harris, S.R., Berriman, M., Parkhill, J., McQuillan, J.A.: Artemis: an integrated platform for visualization and analysis of high-throughput sequence-based experimental data. *Bioinformatics* **28**, 464–469 (2012)