26th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing

# ParallNormal: an efficient variant calling pipeline for unmatched sequencing data

Laura Follia[1,2,3], Fabio Tordini[1], Simone Pernice[1],
Greta Romano[1], Giulia Beatrice Piaggeschi[1,4] and Giulio Ferrero[1,5]

***Abstract*—Nowadays, next generation sequencing is closer to clinical application in the field of oncology. Indeed, it allows the identification of tumor-specific mutations acquired during cancer development, progression and resistance to therapy. In parallel with an evolving sequencing technology, novel computational approaches are needed to cope with the requirement of a rapid processing of sequencing data into a list of clinically-relevant genomic variants.**

**Since sequencing data from both tumors and their matched normal samples are not always available (unmatched data), there is a need of a computational pipeline leading to variants calling in unmatched data. Despite the presence of many accurate and precise variant calling algorithms, an efficient approach is still lacking. Here, we propose a parallel pipeline (*ParallNormal*) designed to efficiently identify genomic variants from whole-exome sequencing data, in absence of their matched normal. ParallNormal integrates well-known algorithms such as BWA and GATK, a novel tool for duplicate removal (*DuplicateRemove*), and the *FreeBayes* variant calling algorithm. A re-engineered implementation of *FreeBayes*, optimized for execution on modern multi-core architectures is also proposed.**

**ParallNormal was applied on whole-exome sequencing data of pancreatic cancer samples without considering their matched normal. The robustness of ParallNormal was tested using results of the same dataset analyzed using matched normal samples and considering genes involved in pancreatic carcinogenesis. Our pipeline was able to confirm most of the variants identified using matched normal data.**

## I. Introduction

Next Generation Sequencing (NGS) is extensively used in the field of oncology to identify genetic variants that leads to differences in an individual's phenotype, trait or risk of developing a disease [1]. Indeed, NGS is applied to identify genetic variants in cancer, in research settings and increasingly in clinical settings for molecular diagnostics and therapy decision. Genetic variants can include Single Nucleotide Polymorphisms (SNPs), difference in a single base pair from a reference, insertions and deletions (indels) of multiple nucleotides, and/or structural variants including, copy number variants, inversions and translocations.

In parallel with an evolving sequencing technology, novel computational approaches are needed to cope with the requirement of a rapid processing of sequencing data into a list of clinically-relevant genomic variants (variant calling). Variant calling tools are widely used with the aim of identifying genetic variants using NGS data [2]. The most used class of variant calling tools is called *probabilistic* methods. These methods use a genotype likelihood framework based on Bayesian probability approach.

Bayes' Theorem describes the probability of each genotype being the correct genotype considering the analyzed data (NGS reads), in terms of the prior probabilities of each possible genotype, and the probability distribution of the data taking into account each possible genotype. Prior information, such as patterns of linkage disequilibrium (i.e. a measure of how often two alleles or specific sequences are inherited together), are joined with other information such as errors in base calling and alignment score, to provide a statistical measure of uncertainty.

Some probabilistic variant calling tools are defined as *haplotype-based* approaches, since the genotype likelihood is estimated by considering different genomic variants mapped in a specific genomic region of interest as not independent from each other.

To properly identify tumor-specific variants most of these tools rely on a matched analysis between tumor and normal samples. However, sequencing data from both tumors and their matched normal samples are not always available (unmatched data), usually due to the lack of biological samples. To cope with this problem, biologists relied on public mutation databases or alternatively they used in-house normal genomes to filter the set of variants which are specific of a tumor sample. Currently, mutation databases have been updated and they contains a huge amount of mutation data giving rise to the necessity of tools and software that support researchers in filtering mutations [3]. For this reason there is a need of a computational pipeline which is not only efficient in the analysis of data but also leading to variants calling user tumor data without associated normal tissue.

Nowadays, several variant calling tools are available to analyze tumor samples without the corresponding normal samples. Genome Analysis Toolkit (GATK) [4], SAMtools [5] and FreeBayes [6], for example, are based on a probabilistic approach and they are able to analyze NGS-data to derive genetic variants from the sample under analysis (e.g. tumor sample) without the healthy counterpart.

Several studies have been conducted in order to evaluate tool performance w.r.t. their ability to call SNPs and short indels with allelic frequencies as low as 1% in matched or non-matched NGS data [7], [8]. However they reported several disagreements among variant calls made by different pipelines, suggesting a need for more careful interpretation of called

[1]Computer Science Department, University of Turin, Italy
[2]Center for Experimental Research and Medical Studies (CeRMS), Italy
[3]Molecular Biotechnology and Health Sciences Department, University of Turin Italy
[4]Italian Institute for Genomic Medicine (IIGM), Italy
[5]Clinical and Biological Sciences Department, University of Turin, Italy

0-7695-6380-5/18/$31.00 ©2018 IEEE
DOI 10.1109/PDP2018.2018.00074

variants.

Moreover, sequencing raw data need to be pre-processed and corrected before variant calling analysis [9]. In particular it is necessary to remove duplicates from the raw reads, map the reads to the reference genome through mapping algorithm and correct read alignments with specific tool. Technically, variant calling is the last and more critical step that needs to be more precise and accurate as possible. For these reasons, the output of a variant calling tool is strictly dependent from the pre-processed input data.

Despite different algorithms having been proposed to identify variants in absence of normal control [3], [10], at the best of our knowledge none of these tools is integrated in a complete analysis pipeline with pre- and post-processing steps of the raw data.

In this paper we present a novel computational pipeline called *ParallNormal*, designed for the first time to efficiently identify a set of genomic variants in unmatched samples. In Section II we describe the main computational modules composing the pipeline. In Section III we highlight the key features required to derive an easy porting of the variant calling module on a multi-core platform. In Section IV we discuss the application of ParallNormal on Whole-Exome Sequencing (WES) data of pancreatic cancer samples, without considering their matched normal. Section V presents the results of the whole pipeline while Section VI provides an evaluation of our pipeline, discusses the obtained results and concludes this work.

## II. Methods

Haplotype-based variant detection methods identify germline variations in and individual's genome. Germline variants are DNA variations coming from germ cells (ovum and sperm) and they could be inherited by the offspring. Very often, these variants occur with a certain frequency throughout the population — as it is the case for SNPs — and require determining the individual's genotype at each genomic region. This process normally involves a number of steps, among which: *1)* pre-processing NGS reads; *2)* aligning reads to a reference genome; *3)* likelihood of variation estimation at each genomic region; *4)* filtering results and SNP annotation.

As reported in Figure 1, ParallNormal adheres to this schema, and is composed of four main analysis modules:

1) reads pre-processing;
2) reads alignment and alignment correction;
3) variant calling;
4) variant filtering.

### A. Reads preprocessing module

Sequencing reads are quality controlled using FastQC[1] and de-duplicated using an in-house algorithm called *DuplicateRemove*. Read de-duplication is required to avoid biases, which are caused by the non-uniform rate of sequence amplification by PCR reaction, performed during NGS library preparation.

[1]http://www.bioinformatics.babraham.ac.uk/projects/fastqc

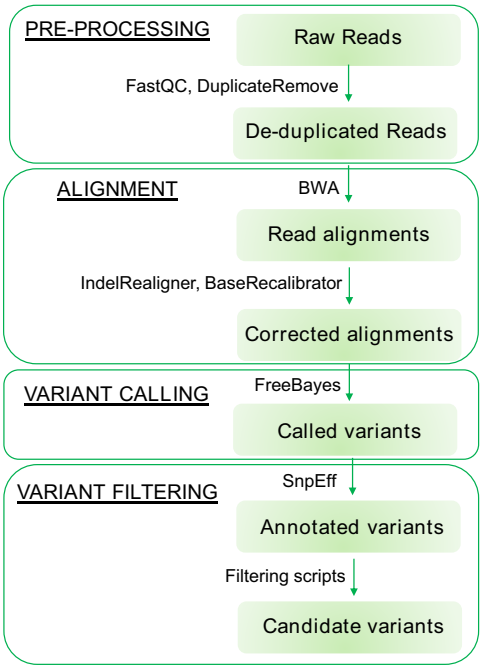


Fig. 1. Flow chart representing the ParallNormal analysis pipeline with a detail on its analysis modules. On each arrow, the algorithms applied in each analysis step are reported.

*DuplicateRemove* applies a filtering step by comparing the k-mer sub-sequences composing each input read, and reads composed of the same k-mer composition are removed. The effect of *DuplicateRemove* on read alignment rates was verified by comparing the fraction of reads aligned before and after the filtering step (Figure 2). As reported in the box plot depicted in the Figure 2, using the non-parametric Wilcoxon signed-rank test no statistically significant differences were found between the number of reads before and after removing duplicates with a $p - value = 0.4287$ for mapped reads and a $p - value = 0.5145$ for properly paired reads.

### B. Reads alignment module

In this module pre-processed reads are aligned against the reference genome using BWA [11]. BWA is a Burrows-Wheeler Transform (BWT) based method that uses a string matching approach to create a space-efficient index of the reference genome to facilitate rapid searching. BWA was selected since it outperformed similar algorithms in variant calling pipeline [8]. Each read alignment is converted in BAM (Binary Alignment/Map) format, sorted, and indexed using SAMtools [5].

Before the variant calling phase, read alignments are processed to avoid biases and false positive variant calls. Initially, reads groups and sample information were added using Picard *AddOrReplaceReadGroups* with standard parameters. Picard is a set of command line tools designed to manipulate sequencing data and formats (i.e. BAM files). This step is performed to

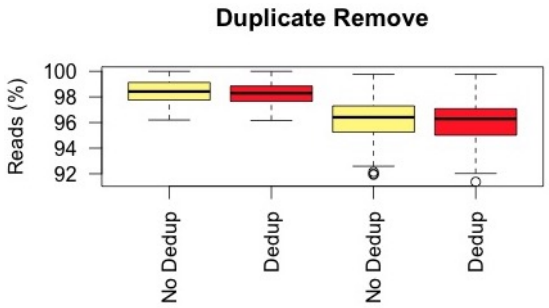


Fig. 2. Box plots illustrating the comparison between mapped and properly paired reads before and after removing duplicates.

know if certain reads were sequenced together on a specific lane, in order to compensate for technical variability among sequencing runs. In this way, all reads within a read group are assumed to come from the same NGS run and share the same error model. Then, read alignments are corrected using three different tools implemented in GATK [4]: *RealignerTargetCreator*, *IndelRealigner*, and *BaseRecalibrator*.

*RealignerTargetCreator* and *IndelRealigner* are applied since misalignment in proximity of indels is a source of error during variant calling. Despite reads may be aligned during the alignment step, their position may be shifted due to the presence of an indel. This shift can introduce false-positive calls in the region close to the variant. For this reason *RealignerTargetCreator* is used to create a subset of reads which are mapped in this context and then a local realignment is performed by *IndelRealigner* in order to minimize the number of mismatching bases. The base quality score recalibration is performed because the estimation provided by the sequencing machines are often inaccurate and it may not reflect the true base-calling error rate. Then, *BaseRecalibrator* implemented in GATK is applied to assign an empirically accurate error model to each bases of the aligned reads. The output of this second correction step is a BAM file ready for the variant calling phase.

### C. Variant calling module: FreeBayes

FreeBayes [7] is a Bayesian genetic variant detector designed to find SNPs, indels, and complex events. It is a haplotype-based variant detector, and uses the actual sequences of reads aligned to a target genomic region to call variants. FreeBayes uses short-read alignments for any number of individuals from a population, and a reference genome to determine the most-likely combination of genotypes for the population at each position in the reference. It then reports resulting variants in Variant Call File (VCF) format.

From a mathematical point of view, FreeBayes is based on a Bayesian model that estimates the probability to find variants at a given genomic region. Specifically, *Bayesian inference* methods produce probabilistic classifications that combine new information derived from the observations with preexisting models (*priors*) that are derived from knowledge. When Bayesian models are applied to genomics, the genome that we are interested in is modeled to derive allele frequencies and prior estimates of the distribution of genotype, and then incorporate read evidence from the observations by estimating how likely it is that a given set of reads is derived from each one of our potential genotypes, for each sample.

The basic idea follows directly from the *Bayes* theorem:

$$\mathbb{P}(Genotype|Data) = \frac{\mathbb{P}(Data|Genotype)\mathbb{P}(Genotype)}{\mathbb{P}(Data)}. \tag{1}$$

Where:

- $\mathbb{P}(Genotype|Data)$ is the probability to obtain a genotype given a set of observations;
- $\mathbb{P}(Data|Genotype)$ explains how likely it would be to see a given set of observations given a particular underlying genotype;
- $\mathbb{P}(Genotype)$ represents the prior likelihood of observing a specific genotype combination;
- $\mathbb{P}(Data)$ is the probability to see a given set of observations.

In this section we will briefly show the mathematical foundations behind this algorithm. We refer to [6] for a more detailed study on how these probabilities are calculated.

At a given genomic region, we have $n$ samples drawn from a population, each of which has a genotype $G_i$ comprised of $k_i$ distinct alleles $b_{i_1}, \ldots, b_{k_i}$. Considering a set of $s_i$ sequencing observations $r_{i_1}, \ldots, r_{i_{s_i}} = R_i$ for each sample $i = 1, \ldots, n$, then to genotype the samples we could apply a Bayesian statistic relating $\mathbb{P}(G_i|R_i)$ to the likelihood of sequencing errors in analized reads, and the prior likelihood of specific genotypes. So the equation 1 becomes

$$\mathbb{P}(G_i|R_i) = \frac{\mathbb{P}(R_i|G_i)\mathbb{P}(G_i)}{P(R_i)}. \tag{2}$$

This means that given a set of sequencing observations and data likelihoods $\mathbb{P}(R_i|G_i)$, for each sample $i$, and the possible genotype derived from the putative alleles, we are able to determine the probability of variants at the considered genomic region. Specifically, a gradient ascent method is employed to determine the maximum a posteriori estimate of a genotyping over all samples under analysis, and to establish an estimate of the probability that there exist variants in the genomic region.

This process begins at the genotyping across all samples $G_1, \ldots, G_n = \{G\}$, where each sample genotype is the maximum likelihood genotype given the data likelihood $\mathbb{P}(R_i|G_i)$. Then the algorithm starts to iterate starting from the genotyping $\{G\}$ estimated in the previous step, and attempts to find a genotyping $G'_1, \ldots, G'_n = \{G'\}$ such that:

$$\mathbb{P}(\{G'\}|R_1, \ldots, R_n) > \mathbb{P}(\{G\}|R_1, \ldots, R_n),$$

and this search iterates until the convergence is reached.

In the end, provided a maximum a posteriori estimate of the genotyping of all samples, it is established an estimate of the quality of the genotyping. To do that, it is estimated the

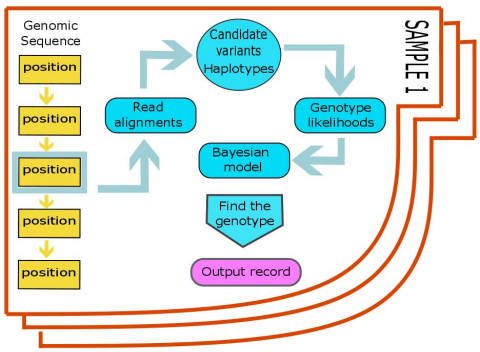


Fig. 3. For each position in the reference genome, FreeBayes detects candidate variants and determines genotypes over a set of samples

probability that the number of distinct alleles at the genomic region $K$ is greater than 1, that is:

$$\mathbb{P}(K > 1|R_1, \ldots, R_n).$$

As exemplified in Figure 3, for each position in the reference genome FreeBayes looks for target read alignments, detects candidate variants and determines all possible genotype alleles for a set of samples. Then, using the Bayesian inference method described above, it derives prior estimates of the distribution of genotypes and allele frequencies, and then incorporate read evidence by estimating how likely a given set of reads is derived from each one of our potential genotypes.

Benchmark analysis showed that FreeBayes performs well in a variant calling process, in absence of paired controls [7], but the algorithm is strongly affected by an overall poor performance, in terms of computational costs and analysis time.

FreeBayes is written as a C++ sequential application, which does not take advantage of the computational power of modern multi-core architectures. It uses genome indexes for direct access to genomic regions, and properly handles aligned reads using external libraries, which helps maintaining a low memory footprint. This is a valuable feature considering the size of the input datasets, that reach several Gigabytes in our experiments.

Authors proposed a workaround to improve such performance bottleneck: a wrapper that permits to run several FreeBayes instances, in parallel, over smaller equally-sized genome regions. This solution produces a sort of *Map+Reduce* behavior, where partial results are merged into a single final output.

While this approach helps reducing the execution time, we found it quite cumbersome because it makes use of external tools and hard-coded scripts to manage genome partitioning and task execution. Nonetheless, computational time is still considerable, the wrapping scripts are not easily portable and require some fixings in order to be usable.

FreeBayes work-flow can be modeled as a *pipeline pattern*, i.e. a functional partitioning of a sequential code that is divided into multiple steps (or *stages)*, executed concurrently onto different consecutive items of an input stream: tasks can start, run, and complete in overlapping time periods. In this way, the whole process can be largely optimized by transforming the search for candidate variants at every location into a stream of locations.

*Stream parallelism* is a programming paradigm supporting the parallel execution of a stream of tasks by using a series of sequential or parallel stages. A stream program can be naturally represented as a graph of independent stages (kernels or filters) that communicate over data channels.

Given a sequence $x_1, \ldots, x_k$ of input tasks, and a simple form of a pipeline with three stages, the computation on each single task $x_i$ is expressed as the composition of three functions $f$, $z$ and $g$, where the second stage (function $z$) works on the results of the application of the first stage, $z(f(x_i))$, and the third stage applies the function $g$ on the output of the second stage: $g(z(f(x_i)))$.

In the general form, a pipeline with stages $s_1, \ldots, s_m$ computes the output stream:

$$s_m(s_{m-1}(\ldots s_2(s_1(x_k))\ldots)), \\ \ldots, s_m(s_{m-1}(\ldots s_2(s_1(x_1))\ldots))$$

Once a pipeline pattern is set up, more complex and interesting behaviors (such as data parallel patterns or tasks farming) can be nested as pipeline stages, which could lead to better exploitation of the computing resources, maximizing efficiency and drastically improving performance. For instance, the steps that compose FreeBayes' procedure express both sequential and *SIMD* (Single Instruction Multiple Data) behaviors, which can be modeled by nesting the proper patterns into the pipeline stages. In Section III we describe our idea to tackle the speeding up of the whole application.

### D. Variant filtering module

The variant filtering step was performed as reported in [7]. Since the variants identified by the variant calling phase are not annotated with respect to genes annotations, SNPeff [12] is applied to annotate each variant. This algorithm provides to each variant different useful information about their position within a gene, discriminating by each protein-coding or non-coding. Furthermore, for variants mapped in protein-coding sequences, this tool provides information about the variant effect on the protein sequence and classifies each variant accordingly. In the filtering module, annotated variants mapped in non-coding regions (UTR, intergenic, intronic, or gene up/down stream regions) were excluded. Mutation not affecting the protein sequence (synonymous mutations) were also excluded. Furthermore, all variants with reads supporting the alternate allele $< 5$ or reads coverage $< 30$ were removed.

## III. Refactoring FreeBayes

After profiling the application, we realized that most of the execution time is devoted to the search of alleles on genomic

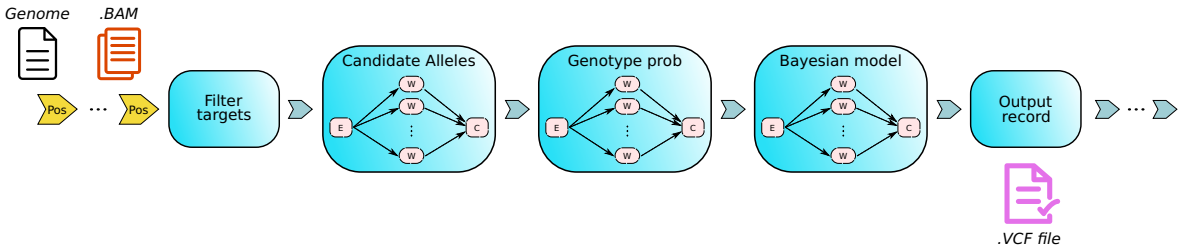


Fig. 4. FreeBayes modeled as a pipeline, whose stages may be task farms or sequential filters. Farms benefit from an emitter function that applies an on-demand scheduling strategy to reduce the load unbalance, due to the high rate of discarded tasks at each step.

positions where target reads alignments are found. This stage becomes more critical when no actual target is given, because the whole genome has to be scanned in chunks in order to find candidate variants.

FreeBayes is characterized by a considerable number of branchings and nested loops, that repeatedly browse and update data structures, and perform frequent I/O operations on input datasets, namely genomic reference files and aligned reads: this design limits the benefits that could derive from compiler optimizations, thus hindering branch predictions and loop nest optimizations that would help to take advantage of data locality and hardware parallelism, provided by internal hardware accelerators such as SSE/MMX.

In order to overcome this problem, we advocate a parallelization schema that supports the concurrent processing of different genomic regions, exploiting the computational power of modern multi-core architectures for improving the overall application performance. We deploy our solution using the FastFlow framework [13], [14], that natively supports high-level parallel programming patterns for working both on data streams and static datasets, and exhibits an efficient *lock-free* run-time support.

In order to achieve our goal, a deep re-engineering of the code is required: we identified those phases that express a data parallel behavior (such as loops where no data dependency exists among processed items), and used FastFlow's `ParallelFor` to introduce loop-level parallelism when possible. This was possible by creating temporary thread-local data structures, that help avoiding dangerous race conditions that may happen when processing shared data structures in parallel.

A pipeline pattern can be employed to model the consecutive steps of candidate alleles identification on each target found in the aligned sequencing reads. Figure 4 illustrates this behavior: chunks of genomic regions of fixed size are dispatched to the very first stage of the pipeline, where they are processed in order to find target aligned reads. Successful targets are forwarded to the next stage, which is responsible for finding candidate alleles on the given targets. The elected tasks advance to the following stages: candidate genotype are processed through the Bayesian model, and their result is written to the output file.

The pipeline is designed so that each stage is composed by a *farm* pattern, that permits to process in parallel different items from the input stream. For instance, a farm stage is equipped with an *emitter* function that builds "windows" of items — stores items in a buffer up to a defined amount — and dispatches them to the workers, where each worker processes them in batch. The output of the farm passes through a *collector*, which may further decorate the results of the workers and forwards the results to the following stage of the pipeline.

Since the ratio of the actually processed sites over the total number of sites is generally rather small, load unbalance among the workers is a concrete risk: the application exhibits an irregular behavior in space and time, because for each position there can be a different number of candidate alleles and selected genotypes, that require different number of iterations. Follows that the parallelization should support the dynamic and active balancing of workload across the involved cores. The emitters of the farms attempt to reduce the load unbalance issue by applying an *on-demand* scheduling policy.

An important factor here is the appropriate concurrency degree for the farm stages, which determines the number of worker threads in each farm, and affects the overall concurrency degree. In general the sum of all the threads concurrently active (pipeline stages + farms' workers) should not exceed the number of available cores. We took care of this requirement by automatically adjusting the worker threads according to the

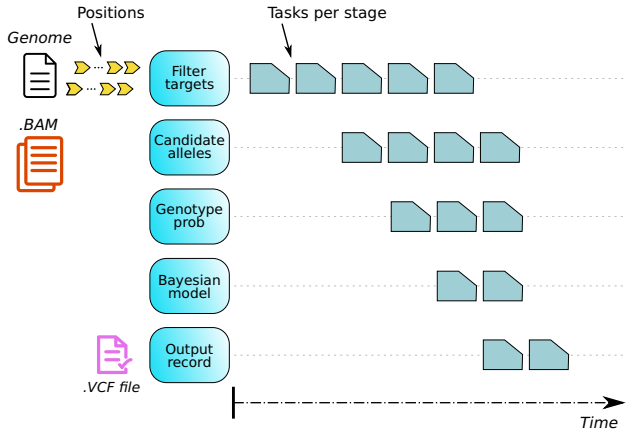


Fig. 5. Concurrent processing of multiple chunks of genomic regions: while tasks advance through the pipeline stages, upcoming items from the input streams are processed.

actual physical CPUs available in the underlying machine.

The execution time of every single phase involved in FreeBayes' pipeline is far from being computationally expensive: the whole application is prevalently an I/O bound problem, due to frequent accesses to datasets and the continuous updates of data structures, and the overall execution time grows linearly with the size of the inputs. The pipeline pattern does not explicitly solve this problem, but permits to efficiently exploit the full computation capabilities of the underlying platform by working concurrently on several genomic regions. This solution concretely helps in reducing the execution time.

In Figure 5, while tasks advance through the pipeline stages, upcoming genome positions (i.e., fixed size chunks) from the genomic reference are dispatched to the first stage of the pipeline and processed, overlapping the execution of other stages: while the first stage filters the input items until it finds target aligned reads, the second stage buffers successful targets and dispatches them to workers for finding candidate alleles on the given targets. Likewise, the subsequent stages operate over previous stages output, until it writes positions putatively polymorphic in Variant Call File (VCF) format.

## IV. Variant detection on WES data

ParallNormal was applied on 10 Whole-Exome Sequencing (WES) from PRJNA289550[2]. These data were generated from experiments performed on Pancreatic Ductal AdenoCarcinoma (PDAC) samples, lacking of matched normal tissue samples. Fastq files were downloaded from European Nucleotide Archive (ENA), study accession PRJNA289550.

### A. Analysis control

As positive control for the analysis, we considered the mutations of three genes well-known to be altered in PDAC: *KRAS*, *TP53*, and *SMAD4*. For each gene the two most common mutations associated to PDAC were used by selecting the annotations of COSMIC v83 database [15].
The set of variants identified by analyzing the same datasets using the matched normal samples were retrieved from *CBioPortal* [16] and used for the comparison of our results.

| ID | Gene name | Protein mutation | Genomic mutation | Genomic Position |
|---|---|---|---|---|
| KRAS_1 | KRAS | p.G12D | c.35G>A | 12:25245350..25245350 |
| KRAS_2 | KRAS | p.G12V | c.35G>T | 12:25245350..25245350 |
| TP53_1 | TP53 | p.R175H | c.524G>A | 17:7675088..7675088 |
| TP53_2 | TP53 | p.R273H | c.818G>A | 17:7673802..7673802 |
| SMAD4_1 | SMAD4 | p.R361H | c.1082G>A | 18:51065549..51065549 |
| SMAD4_2 | SMAD4 | p.R361C | c.1081C>T | 18:51065548..51065548 |

TABLE I
LIST OF THE GENOMIC VARIANTS ANALYZED

## V. Results

To test the efficiency of our pipeline in the variant calling in unmatched samples, were run it on 10 WES datasets of pancreatic cancers. The detection of six true positive variants was evaluated by comparing the results obtained using matched samples.

[2]http://www.ebi.ac.uk/ena/data/view/PRJNA289550

ParallNormal analyses were run on a Linux server equipped with 2 12-core AMD Opteron 6176 SE (48 Hyper-Threads) 2.3GHz, with 12 MiB L3 cache and 512 GiB of main memory, running Linux x86_64. We used ten threads for the preprocessing, reads alignment and correction steps. The average dataset sizes was of 6.06 GiB. As reported in [7], the computing time of the FreeBayes analysis was high, with an average of 14.46 hours per sample analyzed.

As reported in Table II, with our pipeline, we were able to detect most of the true positive variants identified in the matched analysis. In only one sample, one *KRAS* mutation was not identified. Furthermore, none false positive calls of the six analyzed variants was observed.

| ID | Matched analysis | ParallNormal (TP) | ParallNormal (FP) | ParallNormal (FN) |
|---|---|---|---|---|
| KRAS_1 | 3 | 3/3 | 0 | 0 |
| KRAS_2 | 5 | 4/5 | 0 | 1 |
| TP53_1 | 0 | 0 | 0 | 0 |
| TP53_2 | 1 | 1/1 | 0 | 0 |
| SMAD4_1 | 1 | 1/1 | 0 | 0 |
| SMAD4_2 | 0 | 0 | 0 | 0 |

TABLE II
NUMBER OF SAMPLES POSITIVE TO A SPECIFIC VARIANTS AS DEFINED BY THE MATCHED ANALYSIS, AND THE UNMATCHED ANALYSIS PERFORMED WITH PARALLNORMAL. FOR THE UNMATCHED ANALYSES THE NUMBER OF TRUE POSITIVE (TP), FALSE POSITIVE (FP), AND FALSE NEGATIVE (FN) VARIANT CALLS ARE REPORTED

## VI. Conclusion

Despite the use of NGS data in clinic studies is more and more diffuse, different challenges have yet to be address [17]. Efficient computational pipelines are nowadays pivotal to fill the gap between NGS data generation and their summarization in information relevant to the clinic. In the definition of an analysis pipeline, the optimization of its accuracy and its efficiency is mandatory to rapidly extract precise NGS-based information about a patient genome.

For clinical purposes, a NGS data-analysis pipeline must be designed based on the biological problem investigated, and it must be accurate and efficient [9]. The ParallNormal pipeline presented in this work was designed to address these characteristics, in the context of unmatched variant discovery analysis of WES data of cancer samples. A limited number of computational approaches exist to identify variants from tumor-only WES data [3], [10]: to the best of our knowledge, none of them is integrated within an analysis pipeline.

By comparing the variants identified with ParallNormal with those from a matched analysis we observed a quite complete overlap with only one false negative call. Conversely, for a subset of patients, the same gene identified as not mutated in the matched analysis was identified as harboring different variants in our analysis. Indeed, despite our filtering strategy allowed us to narrow down the number of candidate tumor variants, further effort is needed to discriminate germline from somatic variants using our pipeline. For this purpose, we aim to implement a fourth integration module to exploit the information about population frequency of a variant that is collected in public genomic databases like *dbSNP* [18]. Using

these data, we will improve the discriminatory power in the identification of tumor-specific variants.

In conclusion, in this work we presented a novel pipeline for variant calling in unmatched samples, showing its efficiency on a set of pancreatic cancer WES data. Our analysis was able to accurately identify cancer-related variants, but it was characterized by poor performances in term of computational costs and execution time. One bottleneck of the analysis was represented by the variant calling step performed with FreeBayes. After a deep study of the FreeBayes source code, we studied a basic refactoring of its algorithm: by streamlining the detection of candidate alleles, we can concurrently calculate the probability of variants at several genomic regions.

The novel version of FreeBayes is a work in progress: the original sequential application needs to be deeply re-designed in light of the studies presented in Section III. While we have some promising preliminary results, the refactoring still requires tunings and further evaluations, which leaves room for further improvements.

Beside this, we are currently implementing a novel filtering method based on integrative analysis of public genomic databases. We plan to test ParallNormal on a largest cohort of samples, and to provide the whole pipeline as a Docker image, that would facilitate its distribution and installation.

## ACKNOWLEDGEMENT

This work has been supported by Lega Italiana per La Lotta contro i Tumori (LILT).

## REFERENCES

[1] S. Chakravorty and M. Hegde, "Gene and variant annotation for mendelian disorders in the era of advanced sequencing technologies," *Annual Review of Genomics and Human Genetics*, no. 0, 2017.

[2] R. Nielsen, J. S. Paul, A. Albrechtsen, and Y. S. Song, "Genotype and snp calling from next-generation sequencing data," *Nature Reviews Genetics*, vol. 12, no. 6, pp. 443–451, 2011.

[3] S. Hiltemann, G. Jenster, J. Trapman, P. van der Spek, and A. Stubbs, "Discriminating somatic and germline mutations in tumor dna samples without matching normals," *Genome research*, vol. 25, no. 9, pp. 1382–1390, 2015.

[4] M. A. DePristo, E. Banks, R. Poplin, K. V. Garimella, J. R. Maguire, C. Hartl, A. A. Philippakis, G. Del Angel, M. A. Rivas, M. Hanna, *et al.*, "A framework for variation discovery and genotyping using next-generation dna sequencing data," *Nature genetics*, vol. 43, no. 5, pp. 491–498, 2011.

[5] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, and R. Durbin, "The sequence alignment/map format and samtools," *Bioinformatics*, vol. 25, no. 16, pp. 2078–2079, 2009.

[6] E. Garrison and G. Marth, "Haplotype-based variant detection from short-read sequencing," *arXiv preprint arXiv:1207.3907*, 2012.

[7] S. Sandmann, A. O. De Graaf, M. Karimi, B. A. Van Der Reijden, E. Hellström-Lindberg, J. H. Jansen, and M. Dugas, "Evaluating variant calling tools for non-matched next-generation sequencing data," *Scientific Reports*, vol. 7, p. 43169, 2017.

[8] S. Hwang, E. Kim, I. Lee, and E. M. Marcotte, "Systematic comparison of variant calling pipelines using gold standard personal exome variants," *Scientific reports*, vol. 5, 2015.

[9] A. S. Gargis, L. Kalman, D. P. Bick, C. Da Silva, D. P. Dimmock, B. H. Funke, S. Gowrisankar, M. R. Hegde, S. Kulkarni, C. E. Mason, *et al.*, "Good laboratory practice for clinical next-generation sequencing informatics pipelines," *Nature biotechnology*, vol. 33, no. 7, pp. 689–693, 2015.

[10] K. S. Smith, V. K. Yadav, S. Pei, D. A. Pollyea, C. T. Jordan, and S. De, "Somvarius: somatic variant identification from unpaired tissue samples," *Bioinformatics*, vol. 32, no. 6, pp. 808–813, 2015.

[11] H. Li and R. Durbin, "Fast and accurate short read alignment with burrows–wheeler transform," *Bioinformatics*, vol. 25, no. 14, pp. 1754–1760, 2009.

[12] P. Cingolani, A. Platts, L. L. Wang, M. Coon, T. Nguyen, L. Wang, S. J. Land, X. Lu, and D. M. Ruden, "A program for annotating and predicting the effects of single nucleotide polymorphisms, snpeff: Snps in the genome of drosophila melanogaster strain w1118; iso-2; iso-3," *Fly*, vol. 6, no. 2, pp. 80–92, 2012.

[13] M. Aldinucci, M. Danelutto, P. Kilpatrick, and M. Torquati, "Fastflow: high-level and efficient streaming on multi-core," in *Programming Multi-core and Many-core Computing Systems* (S. Pllana and F. Xhafa, eds.), Parallel and Distributed Computing, ch. 13, Wiley, 2017.

[14] F. Tordini, M. Drocco, C. Misale, L. Milanesi, P. Liò, I. Merelli, M. Torquati, and M. Aldinucci, "NuChart-II: the road to a fast and scalable tool for Hi-C data analysis," *International Journal of High Performance Computing Applications*, pp. 1–16, 2016.

[15] S. A. Forbes, D. Beare, H. Boutselakis, S. Bamford, N. Bindal, J. Tate, C. G. Cole, S. Ward, E. Dawson, L. Ponting, *et al.*, "Cosmic: somatic cancer genetics at high-resolution," *Nucleic acids research*, vol. 45, no. D1, pp. D777–D783, 2016.

[16] J. Gao, B. A. Aksoy, U. Dogrusoz, G. Dresdner, B. Gross, S. O. Sumer, Y. Sun, A. Jacobsen, R. Sinha, E. Larsson, *et al.*, "Integrative analysis of complex cancer genomics and clinical profiles using the cbioportal," *Science signaling*, vol. 6, no. 269, p. pl1, 2013.

[17] G. Bertier, M. Hétu, and Y. Joly, "Unsolved challenges of clinical whole-exome sequencing: a systematic literature review of end-users? views," *BMC medical genomics*, vol. 9, no. 1, p. 52, 2016.

[18] S. T. Sherry, M.-H. Ward, M. Kholodov, J. Baker, L. Phan, E. M. Smigielski, and K. Sirotkin, "dbsnp: the ncbi database of genetic variation," *Nucleic acids research*, vol. 29, no. 1, pp. 308–311, 2001.