Hadoop-CNV-RF: A Scalable Copy Number Variation Detection Tool for Next-Generation Sequencing Data

Getiria Onsongo
Macalester College
Saint Paul, Minnesota
gonsongo@macalester.edu

Matthew Bower
Molecular Diagnostics Laboratory, MHealth-Fairview
Minneapolis, Minnesota

Ham Ching Lam
Minnesota Supercomputing Institute
Minneapolis, Minnesota

Bharat Thyagarajan
Department of Laboratory Medicine and Pathology,
University of Minnesota
Minneapolis, Minnesota

ABSTRACT
Detection of small copy number variations (CNVs) in clinically relevant genes is routinely being used to aid diagnosis. We recently developed a tool, CNV-RF, capable of detecting clinically relevant CNVs with a high degree of sensitivity. CNV-RF implementation was designed for small gene panels and did not scale to large gene panels. Analyzing large gene panels with several hundred genes routinely failed due to memory limitations on a single computer, and, when successful, analysis took on average over 24 hours, making it impractical for routine use in the clinic. We need a reliable tool capable of accurately identifying clinically relevant CNVs on large gene panels within a more practical time frame.

We have developed Hadoop-CNV-RF, a freely available, scalable, and more user-friendly implementation of CNV-RF capable of rapidly analyzing large datasets. Hadoop-CNV-RF takes advantage of Hadoop, a framework developed to analyze large amounts of data. In its implementation, we demonstrate the feasibility of developing scalable pipelines on Hadoop that integrate popular bioinformatics software developed for usage on traditional single-user computers without the need for special-purpose routines developed for Hadoop. Results show that Hadoop-CNV-RF reduces analysis time on large gene panels from over 24 hours to about 4 hours on a 20 node Hadoop cluster. Additionally, we demonstrate its ability to scale by analyzing a whole-exome dataset with close to a billion reads. Hadoop-CNV-RF has been clinically validated for large gene panels (up to 4800 genes) and is currently being used in the clinic. It is publicly available at: https://github.com/getiria-onsongo/hadoop-cnvrf-public.

KEYWORDS
Next-Generation Sequencing, Hadoop, Copy Number Variation, High-Performance Computing, Amazon Web Services

1 INTRODUCTION
As the cost of sequencing and analysis continues to decrease, use of Next-Generation Sequencing (NGS) data to improve clinical diagnosis for patients with inherited diseases is increasingly commonplace. This is especially the case for patients with non-specific clinical findings [4, 19, 20, 22]. Whole-genome sequencing is still not economical for routine clinical use and whole-exome sequencing data typically generates a large number of false positives and variants of uncertain significance. A recent comparative analysis by Yao et al [30] of commonly used copy number variation (CNV) detection tools in whole-exome sequencing data concludes that CNV detection using whole-exome sequencing data is still immature for use in the clinic due to its low sensitivity and uncertain specificity. Because of the aforementioned challenges of using whole-genome and whole-exome sequencing data, gene panels are still the most common assay in clinical molecular diagnostics laboratories [22].

We recently developed a tool, CNV-RF, capable of detecting clinically relevant CNVs with a high degree of sensitivity. Its implementation was designed for small gene panels and did not scale to large gene panels. Analyzing large gene panels with several hundreds of genes routinely failed due to memory limitations on a single computer, and, when successful, runtime alone for the software took on average over 24 hours. In practice, the complete analysis process took over two days. On a failed run that had to be submitted for re-analysis, this meant having to wait almost a full week before critical clinical decisions could be made. This long turnaround time made CNV-RF unsuitable for routinely analyzing large gene panels in the clinic.

We examined other recently published software [10, 11, 16, 18] to determine if any of them could be used in the clinic for CNV detection on large gene panels instead of CNV-RF. Of those examined, only DECoN [10] had been clinically validated. DECoN, however, had several limitations. It takes as input “analysis ready” aligned reads, requires that samples be analyzed in cohorts, and is
clinically validated for just two genes: BRCA1 and BRCA2. Furthermore, alignment is the most time-consuming step in the analysis. We needed a tool that took as input unaligned sequenced reads in FASTQ format, easily scaled to analyze large datasets quickly, and had been clinically validated for large gene panels.

To address this need, we have re-implemented CNV-RF using Java[24] to take advantage of Hadoop[9] in a new tool we call Hadoop-CNV-RF. Hadoop is a framework that enables programmers without experience in distributed systems to easily scale analysis to large distributed clusters of commodity hardware. Programmers write their programs using the MapReduce framework [7] and the system takes care of details such as submitting jobs, handling machine failures, and communicating between different machines.

A recent review by Roca et al laments the difficulties faced in using most CNV detection pipelines [27]. To make Hadoop-CNV-RF easily usable, we packaged it into a Java ARchive (JAR) file. To launch an analysis, users execute a single command using this JAR file. Additionally, for users such as clinicians who would prefer not to install dependency software, we pre-installed all of Hadoop-CNV-RF’s dependencies into an Amazon Machine Image (AMI) [1]. This AMI is freely available on Amazon Web Services (AWS)[3].

AWS is a cloud computing infrastructure that provides on-demand access to large compute resources [3]. Because of the substantial resources needed to analyze NGS data, most clinical laboratories without resources to maintain their own large computing infrastructures opt to use cloud computing [22]. And even for those with access to such resources, restrictions associated with validating clinical pipelines make cloud computing resources such as AWS a preferred alternative [23].

Finally, users of Hadoop-CNV-RF are not restricted to using a dedicated Hadoop cluster. Hadoop can be installed on a single computer. Users with small datasets can download it and use it for CNV detection as they would any other software. Hadoop-CNV-RF is a freely available, scalable, and user-friendly software that takes as input unaligned sequenced reads in FASTQ format. It has been clinically validated for large gene panels (up to 4800 genes) and is currently being used in the clinic. Results show it reduces analysis time for large gene panels (over 4800 genes) from over 24 hours to about 4 hours and can easily scale to large FASTQ files and whole-exome data.

2 MATERIALS AND METHODS

2.1 Data

We used the TruSightOne Panel (Illumina, San Diego, CA) for targeted resequencing, which enriches 62,000 exons in 4813 genes (target region size, 12 Mb). Sequencing was performed using a 2 x 100 bp paired-end read on a HiSeq2500 (Illumina, San Diego, CA). Five samples (with matched controls) were analyzed on Amazon Web Services using both CNV-RF and Hadoop-CNV-RF with running times compared. The genomic data is part of patients, medical record and protected by the Health Insurance Portability and Accountability Act (HIPAA). Hence this data is not publicly available.

2.2 Implementation

2.2.1 Overview. A detailed description of the algorithm implemented in Hadoop-CNV-RF was previously described in CNV-RF [22]. The method uses a read-depth approach with sex-matched control to correct for variability in coverage. The algorithm has five main steps: pre-processing, segmentation, detection, filtration, and classification. We re-implemented these steps on a Hadoop cluster. Part of the pre-processing step involves aligning reads and generating coverage data in pileup format using third-party software: BWA-MEM[14], Bowtie 2[13], SAMtools[15], and BEDTools[25]. To ensure scalability, we implemented as part of the program a function that splits the input FASTQ files into smaller partitions, pre-processes the partitions independently on different nodes in the cluster using the third-party software, and saves the processed coverage data in shared space. We re-implemented the remaining steps, (except the final classification step) in MapReduce. The classification step was implemented in R[26] using the packages diptest[12, 17] and randomForest[6]. Running time is negligible for the classification step, which is performed on the master node. Below is a high-level description of the main analysis steps.

2.2.2 Alignment and Coverage Computation. To align reads, the input dataset is split into smaller, roughly equal partitions. The default number of partitions is the number of nodes in a cluster. Users can optionally specify the number of partitions. If the input file (FASTQ) has a total of X reads, and the number of partitions is N, the number of reads in a partition (P) is determined by dividing X by N and then rounding the results to the next integer. That is, \( P = \text{ceil}(X/N) \) where ceil is a function that rounds a fraction to the next integer. The input FASTQ file is then split into smaller files by progressively selecting the first P reads. The result is N - 1 FASTQ files with P reads and one FASTQ file with P or fewer reads. These subsets are sent to different worker nodes for alignment and coverage computation. Figure 1 outlines these steps.

Step 1: Generating metadata file to split input dataset

In Step 1, the total number of reads in the input dataset is determined. A tab-delimited metadata text file containing information needed to partition the dataset is generated. Each line in this metadata file contains all information needed to generate one of the partitions. The metadata files together with the input dataset are pushed to the Hadoop Distributed File System (HDFS) for the next step. HDFS is a shared file system that is accessible to all nodes. Saving data to HDFS is an easy and convenient way to distribute data to any of the nodes.

Step 2: Partitioning input FASTQ file

Step 2 partitions the input dataset into smaller partitions. Each partition is generated independently on one of the worker nodes. To launch partitioning tasks on the different worker nodes, a job is launched with the metadata file generated in Step 1 as input. The job is configured to launch an independent task for each partition. Each of these tasks pulls the input FASTQ from HDFS, extracts one of the partitions, and saves it back to HDFS.

Step 3: Aligning partitioned reads

Step 3 aligns the smaller subset of reads to a reference genome. Similarly to Step 2, an alignment job is launched for each of these
partitions. The job is configured to launch an independent alignment task for each subset of reads. Each task downloads the reference genome from shared space, aligns the subset of reads, groups aligned reads based on chromosomes, and saves the results to a folder on the shared file system corresponding to that chromosome. Before the alignment starts, a directory is created in shared space with subdirectories for each of the chromosomes in the reference genome. For humans, 25 subdirectories (chr 1 to chr22, chrX, chrY, and chrM) are created.

**Step 4: Combining aligned reads**
The final alignment step is combining aligned reads from each partition into groups corresponding to chromosomes in the reference genome. A job is launched in one of the worker nodes for each of the chromosomes. Each job retrieves aligned reads for one of the chromosomes, combines the reads into one alignment file (BAM) for reads mapping to that chromosome, and saves the file in HDFS. As depicted in Fig 1, Step 4, using chromosome 1 on the human genome as an example, reads mapping to chromosome 1 in each of the N partitions is retrieved from shared space. These mapped reads are combined into a single alignment file corresponding to chromosome 1 (Sample 1 chr1.bam), which is then saved back into shared space.

**Step 5: Coverage computation**
Once reads mapping to a chromosome are combined into a single alignment file, a job is launched that generates a tab-delimited text file with coverage data (pileup). A separate independent job is launched for each of the chromosomes. Coverage results are saved back onto shared space for subsequent downstream analysis.

### 2.2.3 Re-implementing CNV-RF functionality using MapReduce
Neither the remaining pre-processing steps nor the segmentation,
detection, filtration, or classification steps depended on third-party software. We re-implemented these steps using the MapReduce framework, which is designed to execute jobs on a distributed cluster. MapReduce automatically partitions jobs, sends them to different worker nodes, and combines the results. In using MapReduce, we did not need to divide input datasets into partitions to take advantage of multiple nodes on the cluster as we did with the alignment and coverage computation steps. Figure 2 is a schematic overview of these remaining steps. All but the last step are executed in a distributed fashion using MapReduce on worker nodes. The final classification step happens on the master node. Users have the option of supplying credentials to a MySQL database with training data used by a machine learning classifier in the last step to categorize candidate CNV calls as either false positives or true positives. If database credentials are not supplied, the software expects as part of the input files a tab-delimited text file with training data. Once the analysis is complete, the results are downloaded onto the user computer and the cluster is terminated.

2.2.4 Chaining jobs into a single pipeline. In the implementation, jobs in the pipeline are joined using a Java program (JobControl class). This program chains jobs together and submits them when ready. A job is ready when all other jobs on which it depends complete successfully.

Figure 3 is a graphical outline of the first few jobs chained together using the JobControl class. This program will automatically start two or more jobs concurrently if they are ready and independent of each other. As Figure 3 shows, partitioning of input data for the sample (Partition Sample FASTQ) and its sex-matched control (Partition Control FASTQ) are independent jobs. These jobs are submitted concurrently. Once the partitioning completes, the program submits the next three alignment jobs: "Align Control FASTQ (BWA)", "Align Sample FASTQ (BWA)"; and "Align Sample FASTQ (Bowtie2)". The program will continue submitting jobs until all jobs in the pipeline have completed.

2.2.5 Launching Hadoop-CNV-RF. Users with a Hadoop cluster can simply download a JAR file for Hadoop-CNV-RF and launch it. We provide instructions on the software page for installing its dependencies and performing CNV analysis. For users without access to a Hadoop cluster, a number of commercial cloud providers exist.
that can provide access to one. Amazon Web Services (AWS) is one of the most popular commercial vendors for cloud computing resources. Because we developed this software for usage in the clinic, an important property guiding our implementation was utility to users without a technical background. To make it easily accessible on AWS to any user, we created a preconfigured Amazon Machine Image (AMI) containing the dependency software needed to run Hadoop-CNV-RF and made it publicly available. No installation is needed to use Hadoop-CNV-RF on AWS. Users can simply point to this AMI when starting their Hadoop cluster and will have a system ready for analysis with all necessary dependency software installed.

To launch the pipeline on a commercial cloud such as AWS, users need to upload input data files. For clinical usage, we recommend uploading common data files that do not change between analyses (such as the reference genome) to the cloud. AWS has a cloud platform for storing large amounts of data called Amazon Simple Storage Service (S3) [2]. To launch the pipeline, the user executes a secure shell (SSH) script that takes as input two configuration files. The first contains location information for FASTQ files and cluster configuration information, such as the number and size of computer nodes in the Hadoop cluster. The second contains credential information needed to access common data files on cloud storage (such as S3) together with variables needed by the pipeline (such as where to save the results). The SSH script copies FASTQ files to the cloud and then launches a Hadoop cluster using information provided in the configuration file. It monitors the cluster status every 60 seconds. Once the cluster is up and running, it starts the analysis.

AWS recently added a functionality that enables users to scale a cluster up or down automatically to meet the resource needs of a given program. In the script that launches the cluster, we added functionality to scale up the cluster and add more nodes if storage is running low or if there are jobs that are ready to run and have been waiting for at least 5 minutes. In the configuration file, users specify the minimum and maximum number of nodes for the cluster. When initially launched, the cluster starts with the minimum number of nodes. It will then scale up as needed by adding one compute node at a time. If cluster usage drops, the cluster will automatically scale down to the minimum number of nodes specified by the user.

The script used to launch the analysis monitors the cluster, checking every 60 seconds to see if the analysis has completed. Once the analysis is done, it will download results to the user computer and terminate the cluster. We provide this SSH script together with sample configuration files and detailed instructions on how to modify them for individual use on our software page.

3 RESULTS

3.1 Number of partitions

A key step in Hadoop-CNV-RF that enables scalability is the splitting of input FASTQ files into smaller partitions and aligning the reads concurrently. The optimal number of partitions (N) will vary based on the size of the input dataset and the number of nodes on a cluster. By default, AWS limits the maximum number of nodes on a Hadoop cluster to 20. We conducted an analysis on AWS using a 20-node Hadoop cluster to determine the optimal N when analyzing data from the TruSightOne Panel (Illumina, San Diego, CA), which enriches 62,000 exons in 4813 genes (target region size, 12 Mb) with approximately 83 million paired-end reads. Each node had 8 virtual CPUs (vCPU) and 64 gigabytes (GB) of memory. In the analysis, we progressively increased the number of partitions from 2 to 40, analyzed the sample, and recorded the running time. Figure 4 shows the results. The x-axis represents the number of partitions (N) and the y-axis represents running time in hours. As these results show, the optimal number of partitions (N) when analyzing our TruSightOne Panel dataset on a 20-node cluster is 20. This result guided our decision to use the number of nodes in a cluster as the default value for N. A similar analysis can be performed on a cluster with different specifications to determine the optimal partition number N.

3.2 Scalability and Cost Considerations

The decision to use the JobControl Java class to chain jobs together into a single pipeline was to ensure that independent jobs can execute concurrently. Figure 3 shows a graphical representation of the first few jobs. As Figure 3 illustrates, the first two jobs of partitioning the sample and partitioning its sex-matched control are independent. Figure 5 shows a snapshot of independent jobs running concurrently in Hadoop-CNV-RF. As soon as the first two jobs complete, the next three jobs start running concurrently.

As previously mentioned, to avoid commissioning a large cluster that is underutilized, we take advantage of a functionality recently added to AWS that enables users to scale a cluster up or down automatically to meet compute demand. Figure 6(a) shows the state of the cluster scaling up to meet Hadoop-CNV-RF resource needs. When these resources are no longer needed, the cluster automatically scales down. Figure 6(b) shows the same cluster scaling down when demand for compute resources reduces. With this functionality, the user only pays for resources they use.
3.3 Runtime Analysis

In our clinical application, the limiting factor was not compute resources but time. We were interested in total running time irrespective of the amount of computer resources consumed. Short turnaround times are desirable to support critical decision-making in the clinic. We compared running times between the old (CNV-RF) and new (Hadoop-CNV-RF) implementations of our CNV detection algorithm. Figure 7 shows these results. The old pipeline took an average of 28.55 hours (standard deviation 4.22). The new pipeline took an average of 3.9 hours (standard deviation 0.98). The new implementation was not only an order of magnitude faster but also more robust. Further, this calculation does not account for extended analysis time due to failed runs, which was common using the old pipeline. In practice, average runtimes for the old pipeline were much longer than 30 hours. In addition to comparing running times, we compared identified CNVs and verified CNV calls were consistent between the two pipelines.

3.4 Clinical Validation

We validated Hadoop-CNV-RF to meet clinical standards. 23 samples were each analyzed twice using the pipeline and results examined to ensure CNVs on specific exons were detected both times. These samples included 20 samples with known CNVs (17 heterozygous deletions, 2 copy number gains, and 1 complex CNV). These known CNVs ranged in size from single-exon events to events involving multiple contiguous genes. In all 20 samples, these specific CNVs were detected on both analyses showing 100% concordance. Three additional samples with no CNV were analyzed to demonstrate specificity and all 3 samples yielded normal results. Duplicates were run to demonstrate concordance of calls across instruments (Nextseq and Novaseq) and flow cells (S1 and S2 flow cells on the Novaseq).

This pipeline is currently being used in the clinic, and all potentially pathogenic CNVs identified by the pipeline were confirmed using quantitative polymerase chain reaction (qPCR). Since we started using the pipeline in the clinic, 13 calls have been identified, of which 8 were validated by qPCR and clinically reported, and 2 were CNVs that were documented as common polymorphisms and not validated by qPCR. The remaining 3 calls were considered false positives.

4 DISCUSSION

We recently developed software to detect clinically relevant small CNVs using targeted resequencing data called CNV-RF. A major contribution of this software was its unique use of machine learning to reduce false positive rates. However, CNV-RF routinely failed due to memory limitation on a single computer. Additionally, it took on average over 24 hours to analyze large gene panels. In a clinical setting where it is not uncommon to analyze over 20 samples a week, this long runtime was not practical and the existing pipeline could only analyze 10 to 15 samples in a routine workweek. This resulted in delayed reporting of clinical results to clinicians.
and longer turnaround times that resulted in clinical samples being sent to other laboratories, as we could not meet appropriate clinical expectations with the existing pipelines. To overcome this limitation, we re-implemented CNV-RF in a new software we call Hadoop-CNV-RF. Hadoop-CNV-RF takes as input FASTQ files and can rapidly scale to large gene panels and whole-exome data. We are now able to analyze 20 samples within 2 days.

To demonstrate that Hadoop-CNV-RF can scale to significantly larger datasets, we analyzed a whole-exome dataset with paired-end reads. The sample had approximately 864 million reads (two FASTQ files, each 164GB in size). The sex-matched control had approximately 1.2 billion reads (two FASTQ files, each 243GB in size). No changes were made to the software or cluster configuration. The complete analysis, from uploading FASTQ files to the cloud to getting the results back, took about 14 hours (wall time = 13 hours 49 minutes 7 seconds).

In comparison, the other software considered expected “analysis ready reads” and recommended samples be analyzed in cohorts. XHMM[11] requires that samples be analyzed in cohorts of at least 10, GermlineCNVCaller[5] requires that samples be analyzed in cohorts of at least 30, and DECn[10] recommends using 3 to 5 control samples. We did an analysis to evaluate the feasibility of using these software by aligning and indexing reads for the whole-exome sample and its sex-matched control to get the prerequisite “analysis ready reads”. Preparing reads for the sample took about 10 hours (wall time = 10 hrs 11 minutes 57 seconds) on a machine with 22 CPUs and 55GB of memory. Preparing reads for its sex-matched control took about 21 hours (wall time = 20 hrs 38 minutes 14 seconds) on a machine with 22 CPUs and 55GB of memory. Five samples will likely take almost a week of continuous compute time on the same machine, and hence we did not further consider any of these software.

In addition to developing a scalable CNV detection tool for clinical use, we demonstrate the feasibility of developing a scalable bioinformatics pipeline on a Hadoop cluster without the need for special-purpose routines specifically designed for the Hadoop framework. There have been efforts to develop tools for common bioinformatics tasks on the Hadoop framework. FASTdoop[8], CloudBurst[28], and Hadoop-BAM[21] were developed to manage FASTQ files, map next-generation sequencing data, and manipulate aligned reads (BAM files) on a Hadoop cluster, respectively. While these special-purpose routines provide solutions for some common bioinformatics tasks on a Hadoop cluster, we still lack routines for most of the commonly used bioinformatics tools. For example, to the best of our knowledge, there is no special-purpose Hadoop tool analogous to BedTools[25]. Furthermore, in certain applications, the original implementation of certain tools is needed. For example, our CNV detection method relied on a difference in implementation between short-read aligners BWA-MEM[14] and Bowtie 2[13] to obtain a sample-specific mappability track. Availability of a special-purpose read aligner for the Hadoop framework would not have been sufficient. In developing Hadoop-CNV-RF, we demonstrated that it is possible to take an existing bioinformatics pipeline that uses software developed for traditional single-user computers (such as BWA-MEM[14], Bowtie 2[13], SAMTools[15], and BEDTools[25]) and re-implement it on Hadoop for scalability without the need for special-purpose routines specifically designed for a Hadoop cluster.

It is worth noting that many genomic analysis problems can easily be broken down into smaller independent problems along chromosomes and analyzed in parallel. Computing coverage and removing duplicates are examples of such problems. We take advantage of this property in scaling the steps from sequence alignment to coverage computation that rely on bioinformatics software developed for traditional single computer use. Developers of other bioinformatics pipelines can use this same approach to scale their pipelines.

A major consideration when developing Hadoop-CNV-RF was usability. A challenge in using most published software in the clinic is that they are not designed for users without a technical background. We created an Amazon Machine Image (AMI) with all necessary dependencies installed and provide step-by-step instructions on how to use it on AWS. We also provide scripts to launch the pipeline and detailed instructions on our software page on how to create an AWS account for its use. The pipeline is currently being used by clinicians.

In addition to using our software on commercial cloud computing infrastructure such as AWS, users have the option of using our software on traditional high-performance computing (HPC) resources. It is increasingly common for research universities to maintain a high-performance computing (HPC) cluster for its researchers. For researchers in the United States, a number of government agencies such as the National Science Foundation (NSF) maintain HPC systems such as XSEDE[29] that are available to researchers. We provide instructions on setting up an ephemeral Hadoop cluster and launching Hadoop-CNV-RF. We tested these instructions on an HP Linux distributed cluster on a shared disk with 1800 total cores at the Minnesota Supercomputing Institute that uses the Portable Batch System (PBS) to distribute jobs.

Finally, as well as providing an accurate and reliable means of detecting clinically relevant CNVs, this new fast and user-friendly software has significantly improved our clinical workflows. We no longer need to send samples to other labs as we did with the old pipeline due to long running times. The short turnaround times have resulted in greater satisfaction by testing clinicians.

5 LIMITATIONS

On rare occasions (less than 5% of the time), Hadoop-CNV-RF fails to analyze genes on certain chromosomes when using AWS. This failure is not consistent or reproducible. In all instances, re-running the sample produces expected results. We are yet to determine if this is an issue with our software or with the cloud provider’s infrastructure. To help users determine when this failure has occurred, Hadoop-CNV-RF provides a quality control file that shows average coverage across all chromosomes. An average coverage close to 0 on a chromosome indicates a failure on this chromosome. We provide a script that automatically re-launches an analysis if this failure occurs. We have not observed this failure when using our HPC cluster.

6 ACKNOWLEDGMENTS

The authors would like to thank Daniel Maseda for proofreading the manuscript.
REFERENCES

[1] Amazon. 2020. Amazon Machine Image. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html. Accessed: 2020-05-18.

[2] Amazon. 2020. Amazon Simple Storage Service. https://aws.amazon.com/s3/. Accessed: 2020-05-18.

[3] Amazon. 2020. Amazon Web Service. https://aws.amazon.com/. Accessed: 2020-05-18.

[4] Arunkanth Ankala, Cristina da Silva, Francesca Guiland, Alessandra Ferlini, Lora JH Bean, Christin Collins, Alice K Tanner, and Madhuri R Hegde. 2015. A comprehensive genomic approach for neuromuscular diseases gives a high diagnostic yield. *Annals of neurology* 77, 2 (2015), 206–214.

[5] Mehrdad Bahdai, David I Benjamin, Samuel K Lee, Andrey Smirnov, Aaron Chevalier, Lee Lichtenstein, and Valentins Ruano Rubio. 2017. GATK CNV: copy-number variation discovery from coverage data.

[6] I Breiman. 2006. randomForest: Breiman and Cutler's Random Forests for Classification and Regression. R package version 4.6-12. Software available at URL: https://cran.rproject.org/web/packages/randomForest (2006).

[7] Jeffrey Dean and Sanjay Ghemawat. 2010. MapReduce: a flexible data processing tool. *Commun. ACM* 53, 1 (2010), 72–77.

[8] Amberto Ferraro Petrillo, Gianluca Roscigno, Giuseppe Cattaneo, and Raffaele Giancarlo. 2017. FASTDoop: a versatile and efficient library for the input of FASTA and FASTQ files for MapReduce Hadoop bioinformatics applications. *Bioinformatics* 33, 10 (2017), 1575–1577.

[9] Apache Software Foundation. 2020. Apache Hadoop. https://hadoop.apache.org/. Accessed: 2020-05-18.

[10] Anna Fowler, Shazia Mahamdallie, Elise Ruark, Sheila Seal, Emma Ramsay, Matthew Clarke, Imran Uddin, Harriet Wylie, Ann Strydom, Gerton Lunter, et al. 2016. Accurate clinical detection of exon copy number variants in a targeted NGS panel using DECO.N. *Welcome open research* 1 (2016).

[11] Menachem Fromer and Shaun M Purcell. 2014. Using XHMM software to detect copy number variation in whole-exome sequencing data. *Current protocols in human genetics* 81, 1 (2014), 7–23.

[12] John A Hartigan, Pamela M Hartigan, et al. 1985. The dip test of unimodality. *The annals of Statistics* 13, 1 (1985), 70–84.

[13] Ben Langmead and Steven L. Salzberg. 2012. Fast gapped-read alignment with Bowtie 2. *Nature methods* 9, 4 (2012), 357.

[14] Heng Li. 2013. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv preprint arXiv:1303.3997* (2013).

[15] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Yue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, and Richard Durbin. 2009. The sequence alignment/map format and SAMtools. *Bioinformatics* 25, 16 (2009), 2078–2079.

[16] Jason Li, Richard Lupat, Kaushalya C Amarasinghe, Ella R Thompson, Maria A Doyle, Georgina L Ryland, Richard W Tothill, Saman K Halgamuge, Ian G Campbell, and Kyle L Gorringe. 2012. CONTRA: copy-number analysis for targeted resequencing. *Bioinformatics* 28, 10 (2012), 1307–1313.

[17] Martin Maechler. 2013. Package ‘diptest’.* R Package Version 0.75–5*. R: a language and environment for statistical computing. Vienna, Austria: R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/

[18] Aaron R Quinlan and Ira M Hall. 2010. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics* 26, 6 (2010), 841–842.

[19] R Core Team. 2018. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/

[20] Iria Roca, Lorena González-Castro, Helena Fernández, M Luz Couce, and Ana Fernández-Marmiesse. 2019. Free-access copy-number variant detection tools for targeted next-generation sequencing data. *Mutation Research/Reviews in Mutation Research* (2019).

[21] Michael C Schatz. 2009. CloudBurst: highly sensitive read mapping with MapReduce. *Bioinformatics* 25, 11 (2009), 1363–1369.

[22] John Towns, Timothy Cocke, Myat A Dahan, Ian Foster, Kelly Gaither, Andrew Grimshaw, Victor Hrelodew, Scott Lathrop, Dave Lefka, Gregory D Peterson, et al. 2014. XSEDE: accelerating scientific discovery. *Computing in science & engineering* 16, 5 (2014), 62–74.

[23] Ruen Yao, Cheng Zhang, Tingting Yu, Niu Li, Xuyun Hu, Xixumin Wang, Jian Wang, and Yiping Shen. 2017. Evaluation of three read-depth based CNV detection tools using whole-exome sequencing data. *Molecular cytogenetics* 10, 1 (2017), 30.

[24] ORACLE. 2020. Java 8. https://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2131355.html. Accessed: 2020-05-18.

[25] Aaron R Quinlan and Ira M Hall. 2010. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics* 26, 6 (2010), 841–842.