GigaScience

GenPipes: an open-source framework for distributed and scalable genomic analyses

--Manuscript Draft--

Manuscript Number:	GIGA-D-18-00198R2
Full Title:	GenPipes: an open-source framework for distributed and scalable genomic analyses
Article Type:	Technical Note
Funding Information:	Canarie — Dr Guillaume Bourque National Sciences and Engineering Research Council — Dr Guillaume Bourque Compute Canada — Dr Guillaume Bourque Genome Canada — Dr Guillaume Bourque Canadian Institute for Health Research — Dr Guillaume Bourque Fonds de Recherche du Québec - Santé — Dr Guillaume Bourque
Abstract:	With the decreasing cost of sequencing and the rapid developments in genomics technologies and protocols, the need for validated bioinformatics software that enables efficient large-scale data processing is growing. Here we present GenPipes, a flexible Python-based framework that facilitates the development and deployment of multi-step workflows optimized for High Performance Computing clusters and the cloud. GenPipes already implements 12 benchmarked and scalable pipelines for various genomics applications, including RNA-Seq, ChIP-Seq, DNA-Seq, Methyl-Seq, Hi-C, capture Hi-C, metagenomics and PacBio long read assembly. The software is available under a GPLv3 open source license and is continuously updated to follow recent advances in genomics and bioinformatics. The framework has been already configured on several servers and a docker image is also available to facilitate additional installations. In summary, GenPipes offers genomic researchers a simple method to analyze different types of data, customizable to their needs and resources, as well as the flexibility to create their own workflows.
Corresponding Author:	Mathieu Bourgey, Ph.D. McGill University and Genome Quebec Innovation Centre Montreal, QC CANADA
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	McGill University and Genome Quebec Innovation Centre
Corresponding Author's Secondary Institution:	
First Author:	Mathieu Bourgey, Ph.D.
First Author Secondary Information:	
Order of Authors:	Mathieu Bourgey, Ph.D. Rola Dali robert eveleigh, Master Kuang Chung Chen Louis Letourneau Joel Fillon Marc Michaud Maxime Caron johanna sandoval

Powered by Editorial Manager® and ProduXion Manager® from Aries Systems Corporation

	Francois Lefebvre
	Gary Leveque
	Eloi Mercier
	David Bujold
	Pascale Marquis
	Patrick Tran Van
	David Morais
	Julien Tremblay
	Xiaojian Shao
	Edouard Henrion
	Emmanuel Gonzalez
	Pierre-Olivier Quirion
	Bryan Caron
	Guillaume Bourque
Order of Authors Secondary Information:	
Response to Reviewers:	Dear Editor, Thank you for the opportunity to submit a revised version of the manuscript GIGA-D-18-00198, which addresses the final points raised by the reviewers. Please find our point-by-point response below. New text that has been added to the revised manuscript is shown in red. Response to the Reviewers: Reviewer #1: I thank the authors for taking the time to address my previous comments. I believe the manuscript is much stronger as a result and I have no further comments to add. We thank the reviewer for his constructive criticism that has strengthened the manuscript. Reviewer #2: The authors successfully address various of my and my colleagues requests. However, certain issues remain, which I will list in the following: # Major * In the introduction, the authors say that frameworks like Galaxy can be inconvenient on large scale projects. Why is that? I think such a claim should be support by a detailed reasoning. Frameworks like Galaxy are generally web-based. For large scale projects, uploading large datasets to a platform can be time/resource intensive. In general, when projects get larger, it is more efficient to bring the software to the data and not upload the data to the software location. We have adjusted the text to say: "… such tools can be inconvenient for large scale projects due to having to move sizeable datasets to the platform". * When mentioning that WMSs rarely provide pre-built pipelines ready for production analysis, the authors should also mention that they nevertheless support development

Powered by Editorial Manager® and ProduXion Manager® from Aries Systems Corporation

of such pipelines by the community of users, including linking out to examples like nf-core and github.com/snakemake-workflows.

We have edited the text accordingly:

“It is important to note that GenPipes, as well as several other WMSs, like Nextflow [58] and SnakeMake [59], support community-developed pipelines, however, those have not been included in the comparison.”

* In my previous comment, I mentioned that the feature table is biased. While the authors added the columns suggested, these where only meant as examples. I would have thought that the authors take this as incentive to get a less biased view, which is arguably very hard. However, even when only taking the reviewer comments as a base, there are plenty of other columns which should go into the table. For example, the authors should add "DRMAA support", "status/progress monitoring" as a column. Moreover, the level of cloud support in GenPipes is quite different from what is offered by e.g. Nextflow and Snakemake. There, you have full Kubernetes support, in case of Snakemake even without the requirement of a shared filesystem. Maybe split the cloud column into "basic cloud support" and "kubernetes support".

Our intentions with Table1 was to provide the reader with an overview of the features of several tools in the field but not necessarily an exhaustive list. We did not design the table to be biased towards GenPipes as we only modified one of the most comprehensive tables we found in recent manuscripts (Griffith & Griffith et al.). Based on that initial table and reviewers’ comments, we added 3 features and 1 WMS. Although more could be added, it would also start cluttering the table and make it difficult to extract meaningful information.

“status/progress monitoring” is already included in the table under “Tracking”. We have modified the column name to make it less ambiguous.

Concerning “DRMAA support” and splitting the “cloud support” column into “basic cloud” and “kubernetes support”, we feel that this is highly technical/specific for the average user.

* The installation mechanism for new software tools (outside of what is provided out of the box) (explained here: https://bitbucket.org/mugqic/genpipes/src/master/#markdown-header-modules), seems like manually redoing all the work that is already solved by package managers like conda or container engines like singularity. For example, Bioconda provides a library of over 4000 bioinformatics software packages which can be readily used from any WMS that supports conda, and Biocontainers provides the same for container based deployment (which lacks conda's ability to rapidly compose custom combinations of tools though). In order to make the comparison fair, the feature table should therefore contain two columns called "package-manager-integration" and "container-integration". For an example of what level of integration I am referring to, see https://snakemake.readthedocs.io/en/stable/snakefiles/deployment.html#integrated-package-management and https://www.nextflow.io/docs/latest/conda.html?highlight=conda.

Bioconda offers a collection of packages and not an integrated system and can be quite heavy in memory requirements. Hence, we think that “package-manager-integration” is not necessarily an indication of the strength of the WMS. It is a specific choice, one that offers ease of installation but has its pitfalls as well. GenPipes does not use package managers by design. GenPipes manages its own libraries making sure there is no conflicting libraries in the process. For users who do not want to install GenPipes manually, we offer a Docker container that has also been tested with Singularity. We have updated the GenPipes’ bitbucket documentation to highlight the availability of the GenPipes’ Docker container.

“container-integration” has already been included in the table under the “Cloud/Container” column.

Powered by Editorial Manager® and ProduXion Manager® from Aries Systems Corporation

	* I am pleased to see that GenPipes indeed supports aggregation over many samples. What remains is the question whether the only entity to aggregate over are samples. If so, only over all samples or is it possible to express e.g. an arbitrary grouping of samples? Moreover, what about other properties, e.g. for scanning a parameter space? I suggest to somehow reflect the different ways of aggregation in the feature table, maybe using the terms that I mentioned in my first review. GenPipes is a flexible python framework that aggregates over readsets, samples and other entities, like chromosomes, based on the pipeline. Arbitrary groupings of samples can be defined in pipelines that use design files, like chipseq and rnaseq. Scanning parameter space can be done by adjusting the configuration files. There isn't a set of limited/defined aggregation methods we use; aggregation is used based on each pipeline's needs. The user can refer to the documentation of each pipeline to see what is possible. For user implemented pipelines, there is no restriction on the aggregations possible. We have added the following lines to the text to highlight some of these points: "… GenPipes can aggregate and merge samples as indicated by the readset file." "… Configuration files are customizable, allowing users to adjust different parameters." "… Custom sample groupings can be defined in the design file." # Minor * Please mention in the caption of the feature table that community based workflows are not considered in the comparison. It might otherwise be that readers overlook this in the main text. We have added test to the caption as follows: "Modified from Griffith & Griffith et al. [57]. Note that community-built pipelines are not considered in the Pipelines section of the table." * Figure S1 contains a lot of typos, e.g. "reasdet", which I guess is supposed to be readset? Thank you, we have corrected the typos in Figure S1.
Additional Information:	
Question	**Response**
Are you submitting this manuscript to a special series or article collection?	No
Experimental design and statistics Full details of the experimental design and statistical methods used should be given in the Methods section, as detailed in our Minimum Standards Reporting Checklist. Information essential to interpreting the data presented should be made available in the figure legends. Have you included all the information requested in your manuscript?	Yes
Resources	Yes

Powered by Editorial Manager® and ProduXion Manager® from Aries Systems Corporation

A description of all resources used, including antibodies, cell lines, animals and software tools, with enough information to allow them to be uniquely identified, should be included in the Methods section. Authors are strongly encouraged to cite Research Resource Identifiers (RRIDs) for antibodies, model organisms and tools, where possible. Have you included the information requested as detailed in our Minimum Standards Reporting Checklist?	
Availability of data and materials All datasets and code on which the conclusions of the paper rely must be either included in your submission or deposited in publicly available repositories (where available and ethically appropriate), referencing such data using a unique identifier in the references and in the “Availability of Data and Materials” section of your manuscript. Have you have met the above requirement as detailed in our Minimum Standards Reporting Checklist?	Yes

Powered by Editorial Manager® and ProduXion Manager® from Aries Systems Corporation

Manuscript

Click here to access/download;Manuscript;GenPipes_GigaScience_Resubmis

Click here to view linked References

GenPipes: an open-source framework for distributed and scalable genomic analyses

Mathieu Bourgey[1,2+*], Rola Dali[1,2+], Robert Eveleigh[1,2], Kuang Chung Chen[3,4], Louis Letourneau[1,2], Joel Fillon[5], Marc Michaud[2], Maxime Caron[1,2,5], Johanna Sandoval[6], Francois Lefebvre[1,2], Gary Leveque[1,2], Eloi Mercier[1,2], David Bujold[1,2], Pascale Marquis[1,2], Patrick Tran Van[7], David Morais[8], Julien Tremblay[9], Xiaojian Shao[1,2], Edouard Henrion[1,2], Emmanuel Gonzalez[1,2], Pierre-Olivier Quirion[1,2], Bryan Caron[3,4], Guillaume Bourque[1,2,5*].

[1] Canadian Centre for Computational Genomics, Montréal, QC, Canada.

[2] McGill University and Genome Québec Innovation Center, Montréal, QC, Canada.

[3] McGill HPC Centre, McGill University, Montréal, QC, Canada.

[4] Calcul Québec, QC, Canada.

[5] Department of Human Genetics, McGill University, Montréal, QC, Canada.

[6] Beaulieu-Saucier Université de Montréal Pharmacogenomics Centre, Montréal, QC, Canada.

[7] Department of Ecology and Evolution, University of Lausanne, Lausanne, Switzerland.

[8] Centre de calcul scientifique (ccs) - Université de Sherbrooke, Sherbrooke, QC, Canada.

[9] Energy, Mining and Environment, National Research Council Canada, Montréal, QC, Canada.

+ First Authors

* To whom correspondence should be addressed. Tel: +1(514) 398-7245; Fax: +1(514) 398-1790;
Email: guil.bourque@mcgill.ca or mathieu.bourgey@mcgill.ca

ABSTRACT

With the decreasing cost of sequencing and the rapid developments in genomics technologies and protocols, the need for validated bioinformatics software that enables efficient large-scale data processing is growing. Here we present GenPipes, a flexible Python-based framework that facilitates the development and deployment of multi-step workflows optimized for High Performance Computing clusters and the cloud. GenPipes already implements 12 validated and scalable pipelines for various genomics applications, including RNA-Seq, ChIP-Seq, DNA-Seq, Methyl-Seq, Hi-C, capture Hi-C, metagenomics and PacBio long read assembly. The software is available under a GPLv3 open source license and is continuously updated to follow recent advances in genomics and bioinformatics. The framework has been already configured on several servers and a docker image is also available to facilitate additional installations. In summary, GenPipes offers genomic researchers a simple method to analyze different types of data, customizable to their needs and resources, as well as the flexibility to create their own workflows.

Keywords: genomics; workflow management systems; frameworks; workflow; pipeline; bioinformatics.

INTRODUCTION

Sequencing has become an indispensable tool in our quest to understand biological processes. Moreover, facilitated by a significant decline in overall costs, new technologies and experimental protocols are being developed at a fast pace. This has resulted in massive amounts of sequencing data being produced and deposited in various public archives. For instance, a number of national initiatives, such as *Genomics England* and *All of US*, plan to sequence hundreds of thousands of individual genomes in an effort to further develop precision medicine. Similarly, a number of large initiatives, such as ENCODE [1] and the International Human Epigenome Consortium (IHEC) [2], plan to generate thousands of epigenomics datasets to better understand gene regulation in normal and disease processes. Despite this rapid progress in sequencing, genomics technologies and available datasets, processing and analyses have struggled to keep up. Indeed, the need for robust, open-source and scalable bioinformatics pipelines has become a major bottleneck for genomics [3].

Available bioinformatics tools for genomic data can be categorized into three different groups: 1) analysis platforms/workbenches, 2) workflow management systems (WMS)/frameworks, and 3) individual analysis pipelines/workflows. Platforms of the first type, like Galaxy [4] or DNA Nexus [5], provide a full workbench for data upload and storage, and are accompanied with a set of available tools. While they provide fast and easy user services, such tools can be inconvenient for large scale projects due to having to move sizeable datasets to the platform. In the second type, WMSs such as Snakemake [6], Nextflow [7], BPipe [8], BigDataScript [9] and declarative workflow description languages, such as CWL or WDL are dedicated to providing a customizable framework to build bioinformatics pipelines. Such solutions are flexible and can help in pipeline implementation but rarely provide robust pre-built pipelines which are ready for production analysis. Finally, tools of the third type are individual analysis pipelines for various applications that have been validated and published. These are useful for specific applications but can sometimes be challenging to implement, difficult to modify or scale-up. They have also rarely been tested on multiple computing infrastructures.

Here we present GenPipes, an open-source, Python-based WMS for pipeline development. As part of its implementation, GenPipes includes a set of high-quality, standardized analysis pipelines, designed for High Performance Computing (HPC) resources and cloud environments. GenPipes' WMS and pipelines have been tested, benchmarked and used extensively over the past four years. GenPipes is continuously updated and is configured on several different HPC clusters with different properties. By combining both WMS and extensively validated End-to-End analysis workflows, GenPipes offers turnkey analyses for a wide range of bioinformatics applications in the genomics field while also enabling flexible and robust extensions.

MATERIAL AND METHODS

Overview of the GenPipes Framework

GenPipes is an object-oriented framework consisting of Python scripts and libraries which create a list of jobs to be launched as Bash commands (Figure 1). There are four main objects that manage the different components of the analysis workflow, namely, *Pipeline, Step, Job and Scheduler*. The main object is the "*Pipeline*" object which controls the workflow of the analysis. Each specific analysis workflow is thus defined as a specific *Pipeline* object. *Pipeline* objects can inherit from one another. The *Pipeline* object defines the flow of the analysis by calling specific *"Step"* objects. The *Pipeline* instance could call all steps implemented in a pipeline or only a set of steps selected by the user. Each step of a pipeline is a unit block that encapsulates a part of the analysis (e.g., trimming or alignment). The *Step* object is a central unit object which corresponds to a specific analysis task. The execution of the task is directly managed by the code defined in each *Step* instance; some steps may execute their task on each sample individually while other steps execute their task using all the samples collectively. The main purpose of the *Step* object is to generate a list of "*Job"* objects which correspond to the consecutive execution of single tasks. The *Job* object defines the commands that will be submitted to the system. It contains all the elements needed to execute the commands, such as input files, modules to be loaded, as well as job dependencies and temporary files. Each *Job* object will be submitted to the system using a specific "*Scheduler"* object. The *Scheduler* object creates execution commands that are compatible with the user's computing system. Four different *Scheduler* objects have already been implemented (PBS, SLURM, Batch and Daemon), see below.

GenPipes' object-oriented framework simplifies the development of new features and its adaptation to new systems; new workflows can be created by implementing a *Pipeline* object which inherits features and steps from other existing *Pipeline* objects. Similarly, deploying GenPipes on a new system may only require the development of the corresponding *Scheduler* object along with specific configuration files. GenPipes' command execution details have been implemented using a shared library system which allows the modification of tasks by simply adjusting input parameters. This simplifies code maintenance and makes changes in software versions consistent across all pipelines.

Freely distributed and pre-installed on a number of HPC resources

GenPipes is an open-source framework freely distributed and open for external contributions from the developer community. GenPipes can be installed from scratch on any Linux cluster supporting Python 2.7 by following the available instructions (https://bitbucket.org/mugqic/genpipes/src/master/). GenPipes can also be used via a Docker image which simplifies the setup process and can be used on a range of platforms, including cloud platforms. This allows system-wide installations, as well as local user installations via the Docker image without needing special permissions.

Through a partnership with the Compute Canada consortium (https://www.computecanada.ca), the pipelines and third-party tools have also been configured on 6 different Compute Canada HPC centers. It allows any Canadian researcher to use GenPipes along with the needed computing resources by simply applying to the consortium [10]. To ensure consistency of pipeline versions and used dependencies (such as genome references and annotation files) and to avoid discrepancy between compute sites, pipeline setup has been centralized to one location which is then distributed on a real-time shared file system: the CERN Virtual Machine File System [11].

Running GenPipes

GenPipes is a command line tool. Its use has been simplified to accommodate general users. A full tutorial is available [12]. Briefly, to launch GenPipes, the following is needed:

- A readset file that contains information about the samples, indicated using the flag "-r". GenPipes can aggregate and merge samples as indicated by the readset file.
- Configuration/ini files that contain parameters related to the cluster and the third-party tools, indicated using the flag "-c". Configuration files are customizable, allowing users to adjust different parameters.
- The specific steps to be executed, indicated by the flag "-s".

The generic command to run GenPipes is:

<pipeline>.py -c myConfigurationFile -r myReadSetFile -s 1-X > Commands.txt && bash Commands.txt

Where <pipeline> can be any of the 12 available pipelines and X is the step number desired. Commands.txt contains the commands that the system will execute.

Pipelines that conduct sample comparisons, like ChIP-Seq and RNA-Seq, require a design file that describes each contrast. Custom sample groupings can be defined in the design file. Design files are indicated by the flag "-d". The tumour_pair pipeline requires normal-tumour pairing information provided in a standard CSV file using the "-p" option. For more information on the design file and the content of each file type, please consult the GenPipes tutorial and the online documentation.

When the GenPipes command is launched, required modules and files will be searched for and validated. If all required modules and files are found, the analysis commands will be produced. GenPipes will create a directed acyclic graph (DAG) that defines job dependency based on input and output of each step. For a representation of the DAG of each pipeline, refer to supplementary figures S1-14. Once launched, the jobs are sent to the scheduler and queued. As jobs complete successfully, their dependent jobs are released by the scheduler to run. If a job fails, all its dependent jobs are terminated and an email notification is sent to the user. When GenPipes is re-run, it will detect which steps have successfully completed, as described in section 'Smart relaunch features', and skip them but will create the command script for the jobs that were not completed successfully. To force the entire command generation, despite successful completion, the "-f" option should be added.

RESULTS

GenPipes was first released in 2014. Since then, it has grown to implement 12 pipelines and is currently installed and maintained on 13 different clusters (Figure 2a-b). GenPipes has been actively used for the last four years to quality control and analyze thousands of samples each year (Figure 2c). It has also been used to analyze data for several large-scale projects such as IHEC [2] and eFORGE [13].

Key features of GenPipes

GenPipes' framework has been optimized to facilitate large scale data analysis. Several features make this possible (Figure 2a):

Multiple schedulers

GenPipes is optimized for HPC processing. It can currently accommodate four different types of schedulers:

- *PBSScheduler* creates a batch script that is compatible with a PBS (TORQUE) system.
- *SLURMscheduler* creates a batch script that is compatible with a SLURM system.
- *BatchScheduler* creates a batch script which contains all the instructions to run all the jobs one after the other.
- *DaemonScheduler* creates a log of the pipeline command in a JSON file.

Job dependencies

In order to minimize the overall analysis time, GenPipes uses a dependency model based on input files, which is managed at the *Job* object level. A job does not need to wait for the completion of a previous step unless it is dependent on its output. Jobs thus become active and can be executed as soon as all their dependencies are met, regardless of the status of previous jobs or of other samples. Thus, when a pipeline

is run on multiple samples, it creates several dependency paths, one per sample, each of which completes at its own pace.

Smart relaunch features

Large scale data analysis is subject to failure which could occur due to system failure (e.g. power outage, system reboot, etc...) or user failure (errors in set parameters, or resources). To limit the micro-management and time required to relaunch the pipeline from scratch, GenPipes includes a system of reporting which provides the status of every job in the analysis in order to facilitate the detection of jobs which have failed. Additionally, a relaunch system is implemented which allows restarting the analysis at the exact state before the failure. The relaunch system uses two features: md5sum hash and time stamps. When GenPipes is launched, a md5sum hash is produced for each command. Upon relaunch following a failure, the newly produced hash is compared to that of the completed job to detect changes in the commands. If the hashes are different, the job is relaunched. To detect updates in input files, GenPipes compares the time stamp on the input and output files of already completed jobs. If the date stamp on the input files is more recent than that on the output files then the job is relaunched. If neither the hash code nor the time stamp flag the job to be relaunched then it is considered complete and up-to-date and it will be skipped in the pipeline restart process.

Configuration files

Running large-scale analyses requires a very large number of parameters to be set. GenPipes implements a superposed configuration system to reduce the time required to set-up or modify parameters needed during the analysis. Configuration files, also referred to as “ini” files, are provided among the arguments of the GenPipes command. These files follow the standard INI format, which was selected for its readability and ease of use by non-expert users. Each pipeline reads all configuration files, one after the other, based on a user defined order. The order is of major importance as the system will overwrite a parameter each time it is specified in a new ini file. The system allows the use of the default configuration files provided in GenPipes alone or in combination with user specific configuration files. Configuration files provided with GenPipes are the result of years of experience along with intensive benchmarking. Additionally, several configuration files adjusted for different compute systems or different model organisms are available. The main advantage of this system is to reduce the users’ task; only parameters that need to be modified (e.g system parameters, genomic resources, user specific parameters) have to be adjusted during the set-up phase of the analysis. To track and enable reproducibility, GenPipes always outputs a file containing the final list of parameters used for the analysis.

Choice among multiple inputs

GenPipes represents a series of *Step* objects that are interdependent based on inputs and outputs. Many of the pipeline steps implemented in GenPipes, represent filtering, manipulation or modification of

specific genomics files share common formats (e.g. bam, fastq, vcf). To ensure more flexibility in the analysis, a system of ordered list to be interpreted as input files is used. For a given *Step*, each *Job* can be given a series of inputs. The *Job* will browse its list of possible inputs and will consider them based on the order in the list. The first input file found either on disk or in the overall output list will be chosen as input. The chosen input will determine the dependency of the *Job* to the other *Jobs* in the pipeline. This system is really flexible and allows users to skip specific steps in the pipeline if they consider them unnecessary.

Customizable workflows

Despite the benchmarking and testing made on the standard analysis procedures implemented in GenPipes, some users may be interested in modifying pipelines. In order to make GenPipes more flexible, a *protocol* system is used. The system allows the implementation of different workflows into a single *Pipeline* object. As a result, one can replace specific steps by other user specific ones. In that case, the user will only need to implement these new Steps and define an additional protocol which will use part of the initial Steps and the newly developed ones. As an example, this has been used to incorporate the Hi-C analysis workflow and the capture Hi-C analysis workflow into GenPipes' hicseq pipeline. A flag (-t hic or -t capture) can be used to specify the workflow to be executed. This system has been developed to reduce the amount of work for external users that decide to contribute to code development and to limit the number of Pipeline objects to maintain. This will also allow us to provide multiple workflows per pipeline to appeal to different tool preferences in each field.

Facilitating dependency installation

Genomic analyses require third party tools, as well as genome sequence files, annotation files and indices. GenPipes comes configured with a large set of reference genomes and their respective annotation files, as well as indices for most aligners. It also includes a large set of third party tools. If GenPipes is being installed from scratch on new clusters, automatic bash scripts that download all tools and genomes are included to ease the setup process. These scripts support local installations without the need for super-user privileges. Tools and dependencies are versioned and are loaded by GenPipes in a version-specific manner. This allows different pipelines to use different software versions based on need. It also allows retention of the same parameters and tools for any given project for reproducibility. GenPipes is also provided as a container version for which no dependency installation is required.

Available workflows

GenPipes implements 12 standardized genomics workflows including: DNA-Seq, Tumour Analysis, RNA-Seq, de novo RNA-Seq, ChIP-Seq, PacBio assembly, Methyl-Seq, Hi-C, capture Hi-C, and Metagenomics (Figure 2c). All pipelines have been implemented following a robust design and development

routine by following established gold standards standard operating protocols (SOP). Below we summarize GenPipes' workflows; more details are available in the GenPipes documentation. For more details concerning computational resources used by each pipeline, refer to supplementary Table S1. All workflows accept a bam or a fastq file as input.

DNA-Seq Pipeline:

DNA-Seq has been implemented optimizing the GATK best practices SOPs [14]. This procedure entails trimming raw reads derived from whole genome or exome data followed by alignment to a known reference, post alignment refinements and variant calling. Trimmed reads are aligned to a reference by the Burrows-Wheeler Aligner, bwa-mem [15]. Refinements of mismatches near indels and base qualities are performed using GATK indels realignment and base recalibration [14] to improve read quality post alignment. Processed reads are marked as fragment duplicates using picard mark duplicates [14] and SNP and small indels are identified using either GATK haplotype callers or samtools mpileup [16]. The Genome in a Bottle [17] dataset was used to select steps and parameters minimizing the false positive rate and maximizing the true positive variants to achieve a sensitivity of 99.7%, precision of 99.1% and F1-score of 99.4% (For more details, refer to Supplementary Materials). Finally, additional annotations are incorporated using dbNSFP [18] and/or Gemini [19XX] and quality control metrics are collected at various stages and visualized using MulitQC [20]. This pipeline has two different protocols, the default protocol based on the GATK variant caller, haplotype_caller, ("-t mugqic", Figure 3) and one based on the mpileup/bcftools caller ("-t mpileup", Figure S1). Another pipeline that is optimized for deep coverage samples, dnaseq_high_coverage, can be found in Figure S2.

RNA-Seq Pipeline:

This pipeline aligns reads with *STAR* [21] 2-passes mode, assembles transcripts with Cufflinks [22] and performs differential expression with *Cuffdiff* [23]. In parallel, gene-level expression is quantified using *htseq-count* [24], which produces raw read counts that are subsequently used for differential gene expression with both *DESeq* [25] and *edgeR* [26]. Several common quality metrics (rRNA content, expression saturation estimation etc.) are also calculated through the use of *RNA-SeQC* [27] and in-house scripts. Gene Ontology terms are also tested for over-representation using *GOseq* [28]. Expressed short SNVs and indels calling is also performed by this pipeline, which optimizes GATK best practices to reach a sensitivity 92.8%, precision 87.7% and F1-score 90.1%. A schema of pipeline steps can be found in Figure S3. Another pipeline, rnaseq_light, based on Kallisto [29] and used for quick quality control can be found in Figure S4.

De-Novo RNASeq Pipeline:

This pipeline is adapted from the Trinity-Trinotate suggested workflow [30] [31]. It reconstructs transcripts from short reads, predicts proteins and annotates leveraging several databases. Quantification

is computed using RSEM and differential expression is tested in a manner identical to the RNA-seq pipeline. We observed that the default parameters of the Trinity suite are very conservative which could result in the loss of low-expressed but biologically relevant transcripts. In order to provide the most complete set of transcripts, the pipeline was designed with lower stringency during the assembly step in order to produce every possible transcript and not miss low expressed mRNA. A stringent filtration step is included afterward in order to provide a set of transcripts that make sense biologically. A schema of pipeline steps can be found in Figure S5.

ChIP-Seq Pipeline:

The ChIP-Seq workflow is based on the ENCODE [1] workflow. It aligns reads using the Burrows-Wheeler Aligner. It creates tag directories using Homer [32]. Peaks are called using MACS2 [33] and annotated using Homer. Binding motifs are also identified using Homer. Metrics are calculated based on IHEC requirements [34]. The ChIP-Seq pipeline can also be used for ATAC-Seq samples. However, we are developing a pipeline that is specific to ATAC-Seq. A schema of pipeline steps can be found in Figure S6.

The Tumour Analysis Pipeline:

The Tumour Pair workflow inherits the bam processing protocol from DNA-seq implementation to retain the benchmarking optimizations but differs in alignment refinement and mutation identification by maximizing the information utilizing both tumour and normal samples together. The pipeline is based on an ensemble approach, which was optimized using both the DREAM3 challenge [35] and the CEPH mixture datasets to select the best combination of callers for both SNV and SV detection. For SNVs, multiple callers such as GATK mutect2, VarScan2 [36], bcftools and VarDict [37] were combined to achieve a sensitivity of 97.5%, precision of 98.8% and F1-score of 98.1% for variants found in 2 or more callers. Similarly, SVs were identified using multiple callers: DELLY [38], LUMPY [39], WHAM [40], CNVkit [41] and Svaba [42] and combined using MetaSV [43] to achieve a sensitivity of 84.6%, precision of 92.4% and F1-score of 88.3% for duplication variants found in the DREAM3 dataset (For more details, refer to Supplementary Material). The pipeline also integrates specific cancer tools to estimate tumour purity, tumour ploidy of sample pair normal-tumour. Additional annotations are incorporated to the SNV calls using dbNSFP [18] and/or Gemini [19] and quality control metrics were collected at various stages and visualized using MulitQC [20]. This pipeline has 3 protocols (sv, ensemble or fastpass). Schemas of pipeline steps for the three protocols can be found in Figures S7, 8 and 9.

Whole Genome Bisulfite Seq Pipeline (WGBS or Methyl-Seq):

The Methyl-Seq workflow is adapted from the Bismark pipeline [44]. It aligns paired-end reads with botiwe2 default mode. Duplicates are removed with Picard and methylation calls are extracted using bismark [44]. Wiggle tracks for both read coverage and methylation profile are generated for visualization.

Variants calls can be extracted from the WGBS data directly using bisSNP [45]. Bisulfite conversion rates are estimated with lambda genome or from human non-CpG methylation directly. Several metrics based on IHEC requirements are also calculated. Methyl-Seq can also process capture data if provided with a capture bed file. A schema of pipeline steps can be found in Figure S10.

Hi-C Pipeline:

The HiC-Seq workflow aligns reads using HiCUP [46]. It creates tag directories, produces interaction matrices, identifies compartments and significant interactions using Homer. It identifies Topologically Associating Domains using TopDom [47] and RobusTAD (bioRxiv 293175). It also creates ".hic" files using JuiceBox [48] and metrics reports using MultiQC [20]. The HiC-Seq workflow can also process capture Hi-C data with the flag "-t capture" using CHICAGO [49]. Schemas for the HiC and capture HiC protocols of this pipeline can be found in Figure S11 and Figure S12 respectively.

The Metagenomic Pipeline (rRNA gene amplification analysis):

This pipeline is based on the established Qiime procedure [50] for amplicon-based metagenomics. It assembles read pairs using FLASH [51], detects chimeras with uchime [52] and picks OTUs using vsearch [53]. OTUs are then aligned using PyNAST [54] and clustered with FastTree [55]. Standard diversity indices, taxonomical assignments and ordinations are then calculated and reported graphically. A schema of pipeline steps can be found in Figure S13.

The PacBio Pipeline:

The PacBio whole genome assembly pipeline is built following the HGAP method [31], including additional features, such as base modification detection (https://github.com/PacificBiosciences/Bioinformatics-Training/wiki/Methylome-Analysis-Technical-Note) and genome circularization [56]. De novo assembly is performed using PacBio's SMRT Link software (https://github.com/PacificBiosciences/SMRT-Link/wiki). Assembly contigs are generated using HGAP4. Alignments are then corrected and used as seeds by FALCON (https://github.com/PacificBiosciences/FALCON/wiki/) to create contigs. The resulting contigs are then polished and processed by "Arrow" (https://github.com/PacificBiosciences/GenomicConsensus) which ultimately generates high quality consensus sequences. An optional step allowing assembly circularization is integrated at the end of the pipeline. A schema of pipeline steps can be found in Figure S14.

Comparison with other solutions for NGS analysis

Data collected for select tools modified from Griffith & Griffith et al. [57] (Table 1), shows that GenPipes' strength lies in its robust WMS that comes with one of the most diverse selection of analysis pipelines which have been thoroughly tested. The pipelines in the framework cover a wide range of sequencing applications (Figure 2a). The pipelines are end-to-end workflows running complete bioinformatics analyses. While many available pipelines conclude with a bam file or run limited post-bam analysis steps, the pipelines included in GenPipes are extensive, often having as many as 40 different steps that cover a wide range of post-bam processing. It is important to note that GenPipes, as well as several other WMSs, like Nextflow [58] and SnakeMake [59], support community-developed pipelines, however, those have not been included in the comparison.

GenPipes is compatible with HPC computing, as well as cloud computing [60] and includes a workflow manager that can be adapted to new systems. GenPipes also provides job status tracking through JSON files that can then be displayed on a web portal (an official portal for GenPipes will be released soon). GenPipes' available pipelines facilitate bioinformatics processing, while the framework makes it flexible for modifications and new implementations.

GenPipes developers offer continuous support through a Google forum page [61] and a help desk email address (pipelines@computationalgenomics.ca). Since the release of version 2.0.0 in 2014, a community of users has run GenPipes to conduct approximately 3000 analyses processing around 100,000 samples (Figure 2b-c).

DISCUSSION and CONCLUSION

GenPipes is a workflow management system that facilitates building robust genomic workflows. GenPipes is a unique solution which combines both a framework for development and end-to-end analysis pipelines for a very large set of genomics fields. The efficient framework for pipeline development has resulted in a broad community of developers with over 30 active branches and more than 10 forks of the GenPipes repository. GenPipes has several optimized features that adapt it to large scale data analysis, namely:

- **Multiple schedulers**: GenPipes is optimized for HPC processing. It currently accommodates 4 schedulers.
- **Job dependencies**: GenPipes establishes dependencies among its different steps. This enables launching all the steps at the same time and minimizes queue waiting time and management.
- **Smart relaunch**: GenPipes sets and detects flags at each successful step in the pipeline. This allows the detection of successfully completed steps and easy relaunch of failed steps.
- **Parameter encapsulation**: Genpipes uses a superposed configuration system to parse all required parameters from configuration files. This simplifies the use of the framework and makes it more flexible to user adjustments. Tested configuration files that are tailored to different clusters and different species are included with GenPipes.

- **Diverse inputs**: GenPipes has been developed to launch using different starting inputs, making it more flexible.
- **Flexible workflows**: GenPipes implements a workflow in steps. Users can choose to run specific steps of interest, limiting waste of time and resources.

GenPipes is under continuous development to update established pipelines and to create new pipelines for emerging technologies. For instance, new genomics pipelines are being developed for ATAC-Seq, single cell RNA-Seq and HiChIP. GenPipes is also being redeveloped to use the Common Workflow Language (CWL) to provide a cloud compatible version more seamlessly and more *Scheduler* objects, like DRMAA, are being added to expand compatibility with more platforms. GenPipes has become a reliable bioinformatics solution that has been used in various genomics publications for DNA-Seq [62-69], RNA-Seq [70] and ChIP-Seq [71] analyses. GenPipes is currently available as source code, as well as a Docker image for easy installation and use. GenPipes has been optimized for HPC systems but can run on a laptop computer on small datasets.

Availability and requirements

- Project name: GenPipes
- Project home page: http://www.c3g.ca/genpipes
- Operating system(s): Linux; Can be used on Windows and Mac OS using Docker
- Programming language: Python
- Other requirements: Workflow-dependant; detailed in documentation
- License: GNU GPLv3
- SciCrunch RRID: SCR_016376

SUPPLEMENTARY DATA

No Supplementary Data

ACKNOWLEDGEMENT

Data analyses were enabled by compute and storage resources provided by Compute Canada and Calcul Québec. Authors would also like to acknowledge Romain Gregoire and Tushar Dubey for their contribution to the code and Patricia Goerner-Potvin for her help in planning the report content.

FUNDING

This work was supported by CANARIE, Compute Canada and Genome Canada. Additional support came from a grant from the National Sciences and Engineering Research Council (NSERC-448167-2013) and a grant from the Canadian Institute for Health Research (CIHR-MOP-115090). GB is also supported by the Fonds de Recherche Santé Québec (FRSQ-25348).

CONFLICT OF INTEREST

The Authors declare no conflict of interest.

REFERENCES

1. ENCODE, *The ENCODE (ENCyclopedia Of DNA Elements) Project.* Science, 2004. **306**(5696): p. 636-40.
2. Stunnenberg, H.G. and M. Hirst, *The International Human Epigenome Consortium: A Blueprint for Scientific Collaboration and Discovery.* Cell, 2016. **167**(5): p. 1145-1149.
3. Mardis, E.R., *The $1,000 genome, the $100,000 analysis?* Genome Med, 2010. **2**(11): p. 84.
4. Afgan, E., et al., *The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update.* Nucleic Acids Res, 2016. **44**(W1): p. W3-W10.
5. DNANexus website, *https://www.dnanexus.com/.*
6. Koster, J. and S. Rahmann, *Snakemake--a scalable bioinformatics workflow engine.* Bioinformatics, 2012. **28**(19): p. 2520-2.
7. Di Tommaso, P., et al., *Nextflow enables reproducible computational workflows.* Nat Biotechnol, 2017. **35**(4): p. 316-319.
8. Sadedin, S.P., B. Pope, and A. Oshlack, *Bpipe: a tool for running and managing bioinformatics pipelines.* Bioinformatics, 2012. **28**(11): p. 1525-6.
9. Cingolani, P., R. Sladek, and M. Blanchette, *BigDataScript: a scripting language for data pipelines.* Bioinformatics, 2015. **31**(1): p. 10-6.
10. Compute Canda, *https://www.computecanada.ca/research-portal/account-management/apply-for-an-account/.*
11. P. Buncic, C.A.S., J. Blomer, L. Franco, A. Harutyunian, P. Mato, and Y. Yao., *CernVM - a virtual software appliance for LHC applications*, in *Journal of Physics*. 2010. p. 042003.
12. GenPipes tutorial, *http://www.computationalgenomics.ca/tutorials/.*
13. Breeze, C.E., et al., *eFORGE: A Tool for Identifying Cell Type-Specific Signal in Epigenomic Data.* Cell Rep, 2016. **17**(8): p. 2137-2150.
14. Van der Auwera, G.A., et al., *From FastQ data to high confidence variant calls: the Genome Analysis Toolkit best practices pipeline.* Curr Protoc Bioinformatics, 2013. **43**: p. 11 10 1-33.
15. Li, H. and R. Durbin, *Fast and accurate short read alignment with Burrows-Wheeler transform.* Bioinformatics, 2009. **25**(14): p. 1754-60.
16. Li, H., et al., *The Sequence Alignment/Map format and SAMtools.* Bioinformatics, 2009. **25**(16): p. 2078-9.
17. Zook, J.M., et al., *Extensive sequencing of seven human genomes to characterize benchmark reference materials.* Sci Data, 2016. **3**: p. 160025.
18. Liu, X., et al., *dbNSFP v3.0: A One-Stop Database of Functional Predictions and Annotations for Human Nonsynonymous and Splice-Site SNVs.* Hum Mutat, 2016. **37**(3): p. 235-41.
19. Paila, U., et al., *GEMINI: integrative exploration of genetic variation and genome annotations.* PLoS Comput Biol, 2013. **9**(7): p. e1003153.

20. Ewels, P., et al., *MultiQC: summarize analysis results for multiple tools and samples in a single report.* Bioinformatics, 2016. **32**(19): p. 3047-8.
21. Dobin, A., et al., *STAR: ultrafast universal RNA-seq aligner.* Bioinformatics, 2013. **29**(1): p. 15-21.
22. Trapnell, C., et al., *Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation.* Nat Biotechnol, 2010. **28**(5): p. 511-5.
23. Trapnell, C., et al., *Differential analysis of gene regulation at transcript resolution with RNA-seq.* Nat Biotechnol, 2013. **31**(1): p. 46-53.
24. Anders, S., P.T. Pyl, and W. Huber, *HTSeq--a Python framework to work with high-throughput sequencing data.* Bioinformatics, 2015. **31**(2): p. 166-9.
25. Anders, S. and W. Huber, *Differential expression analysis for sequence count data.* Genome Biol, 2010. **11**(10): p. R106.
26. Robinson, M.D., D.J. McCarthy, and G.K. Smyth, *edgeR: a Bioconductor package for differential expression analysis of digital gene expression data.* Bioinformatics, 2010. **26**(1): p. 139-40.
27. DeLuca, D.S., et al., *RNA-SeQC: RNA-seq metrics for quality control and process optimization.* Bioinformatics, 2012. **28**(11): p. 1530-2.
28. Young, M.D., et al., *Gene ontology analysis for RNA-seq: accounting for selection bias.* Genome Biol, 2010. **11**(2): p. R14.
29. Bray, N.L., et al., *Near-optimal probabilistic RNA-seq quantification.* Nat Biotechnol, 2016. **34**(5): p. 525-7.
30. Grabherr, M.G., et al., *Full-length transcriptome assembly from RNA-Seq data without a reference genome.* Nat Biotechnol, 2011. **29**(7): p. 644-52.
31. Chin, C.S., et al., *Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data.* Nat Methods, 2013. **10**(6): p. 563-9.
32. Heinz, S., et al., *Simple combinations of lineage-determining transcription factors prime cis-regulatory elements required for macrophage and B cell identities.* Mol Cell, 2010. **38**(4): p. 576-89.
33. Zhang, Y., et al., *Model-based analysis of ChIP-Seq (MACS).* Genome Biol, 2008. **9**(9): p. R137.
34. IHEC standards, *https://github.com/IHEC/ihec-assay-standards.*
35. Ewing, A.D., et al., *Combining tumor genome simulation with crowdsourcing to benchmark somatic single-nucleotide-variant detection.* Nat Methods, 2015. **12**(7): p. 623-30.
36. Koboldt, D.C., et al., *VarScan 2: somatic mutation and copy number alteration discovery in cancer by exome sequencing.* Genome Res, 2012. **22**(3): p. 568-76.
37. Lai, Z., et al., *VarDict: a novel and versatile variant caller for next-generation sequencing in cancer research.* Nucleic Acids Res, 2016. **44**(11): p. e108.
38. Rausch, T., et al., *DELLY: structural variant discovery by integrated paired-end and split-read analysis.* Bioinformatics, 2012. **28**(18): p. i333-i339.
39. Layer, R.M., et al., *LUMPY: a probabilistic framework for structural variant discovery.* Genome Biol, 2014. **15**(6): p. R84.
40. Kronenberg, Z.N., et al., *Wham: Identifying Structural Variants of Biological Consequence.* PLoS Comput Biol, 2015. **11**(12): p. e1004572.
41. Talevich, E., et al., *CNVkit: Genome-Wide Copy Number Detection and Visualization from Targeted DNA Sequencing.* PLoS Comput Biol, 2016. **12**(4): p. e1004873.
42. Wala, J.A., et al., *SvABA: genome-wide detection of structural variants and indels by local assembly.* Genome Res, 2018. **28**(4): p. 581-591.
43. Mohiyuddin, M., et al., *MetaSV: an accurate and integrative structural-variant caller for next generation sequencing.* Bioinformatics, 2015. **31**(16): p. 2741-4.
44. Krueger, F. and S.R. Andrews, *Bismark: a flexible aligner and methylation caller for Bisulfite-Seq applications.* Bioinformatics, 2011. **27**(11): p. 1571-2.
45. Liu, Y., et al., *Bis-SNP: combined DNA methylation and SNP calling for Bisulfite-seq data.* Genome Biol, 2012. **13**(7): p. R61.
46. Wingett, S., et al., *HiCUP: pipeline for mapping and processing Hi-C data.* F1000Res, 2015. **4**: p. 1310.
47. Shin, H., et al., *TopDom: an efficient and deterministic method for identifying topological domains in genomes.* Nucleic Acids Res, 2016. **44**(7): p. e70.
48. Durand, N.C., et al., *Juicer Provides a One-Click System for Analyzing Loop-Resolution Hi-C Experiments.* Cell Syst, 2016. **3**(1): p. 95-8.

49. Cairns, J., et al., *CHiCAGO: robust detection of DNA looping interactions in Capture Hi-C data.* Genome Biol, 2016. **17**(1): p. 127.
50. Kuczynski, J., et al., *Using QIIME to analyze 16S rRNA gene sequences from microbial communities.* Curr Protoc Bioinformatics, 2011. **Chapter 10**: p. Unit 10.7.
51. Magoc, T. and S.L. Salzberg, *FLASH: fast length adjustment of short reads to improve genome assemblies.* Bioinformatics, 2011. **27**(21): p. 2957-63.
52. Edgar, R.C., et al., *UCHIME improves sensitivity and speed of chimera detection.* Bioinformatics, 2011. **27**(16): p. 2194-200.
53. Rognes, T., et al., *VSEARCH: a versatile open source tool for metagenomics.* PeerJ, 2016. **4**: p. e2584.
54. Caporaso, J.G., et al., *PyNAST: a flexible tool for aligning sequences to a template alignment.* Bioinformatics, 2010. **26**(2): p. 266-7.
55. Price, M.N., P.S. Dehal, and A.P. Arkin, *FastTree: computing large minimum evolution trees with profiles instead of a distance matrix.* Mol Biol Evol, 2009. **26**(7): p. 1641-50.
56. Hunt, M., et al., *Circlator: automated circularization of genome assemblies using long sequencing reads.* Genome Biol, 2015. **16**: p. 294.
57. Griffith, M., et al., *Genome Modeling System: A Knowledge Management Platform for Genomics.* PLoS Comput Biol, 2015. **11**(7): p. e1004274.
58. NextFlow Community Pipelines, *https://github.com/nf-core.*
59. SnakeMake Community Pipelines, *https://github.com/snakemake-workflows.*
60. GenPipes Cloud, *http://www.computationalgenomics.ca/genpipes-in-the-cloud/.*
61. GenPipes GoogleForum, *https://groups.google.com/forum/#!forum/GenPipes.*
62. Buczkowicz, P., et al., *Genomic analysis of diffuse intrinsic pontine gliomas identifies three molecular subgroups and recurrent activating ACVR1 mutations.* Nat Genet, 2014. **46**(5): p. 451-6.
63. Scelo, G., et al., *Variation in genomic landscape of clear cell renal cell carcinoma across Europe.* Nat Commun, 2014. **5**: p. 5135.
64. Le Guennec, K., et al., *17q21.31 duplication causes prominent tau-related dementia with increased MAPT expression.* Mol Psychiatry, 2017. **22**(8): p. 1119-1125.
65. Torchia, J., et al., *Integrated (epi)-Genomic Analyses Identify Subgroup-Specific Therapeutic Targets in CNS Rhabdoid Tumors.* Cancer Cell, 2016. **30**(6): p. 891-908.
66. Oliazadeh, N., et al., *Identification of Elongated Primary Cilia with Impaired Mechanotransduction in Idiopathic Scoliosis Patients.* Sci Rep, 2017. **7**: p. 44260.
67. Bellenguez, C., et al., *Contribution to Alzheimer's disease risk of rare variants in TREM2, SORL1, and ABCA7 in 1779 cases and 1273 controls.* Neurobiol Aging, 2017. **59**: p. 220.e1-220.e9.
68. Hamdan, F.F., et al., *High Rate of Recurrent De Novo Mutations in Developmental and Epileptic Encephalopathies.* Am J Hum Genet, 2017. **101**(5): p. 664-685.
69. Monlong, J., et al., *Global characterization of copy number variants in epilepsy patients from whole genome sequencing.* PLoS Genet, 2018. **14**(4): p. e1007285.
70. Manku, G., et al., *Changes in the expression profiles of claudins during gonocyte differentiation and in seminomas.* Andrology, 2016. **4**(1): p. 95-110.
71. Deblois, G., et al., *ERRalpha mediates metabolic adaptations driving lapatinib resistance in breast cancer.* Nat Commun, 2016. **7**: p. 12156.
72. Fisch, K.M., et al., *Omics Pipe: a community-based framework for reproducible multi-omics data analysis.* Bioinformatics, 2015. **31**(11): p. 1724-8.
73. Reich, M., et al., *GenePattern 2.0.* Nat Genet, 2006. **38**(5): p. 500-1.
74. O'Connor, B.D., B. Merriman, and S.F. Nelson, *SeqWare Query Engine: storing and searching sequence data in the cloud.* BMC Bioinformatics, 2010. **11 Suppl 12**: p. S2.
75. Buske, F.A., et al., *NGSANE: a lightweight production informatics framework for high-throughput data analysis.* Bioinformatics, 2014. **30**(10): p. 1471-2.
76. Ceraj, I., Riley, J. T., Shubert, C., *StarHPC - Teaching Parallel Programming within Elastic Compute Cloud.* Proceedings of the ITI 2009 31st Int. Conf. on Information Technology Interfaces, June 22-25, 2009.
77. Taghiyar, M.J., et al., *Kronos: a workflow assembler for genome analytics and informatics.* Gigascience, 2017. **6**(7): p. 1-10.

TABLE AND FIGURES LEGENDS

Figure 1 - General workflow of GenPipes

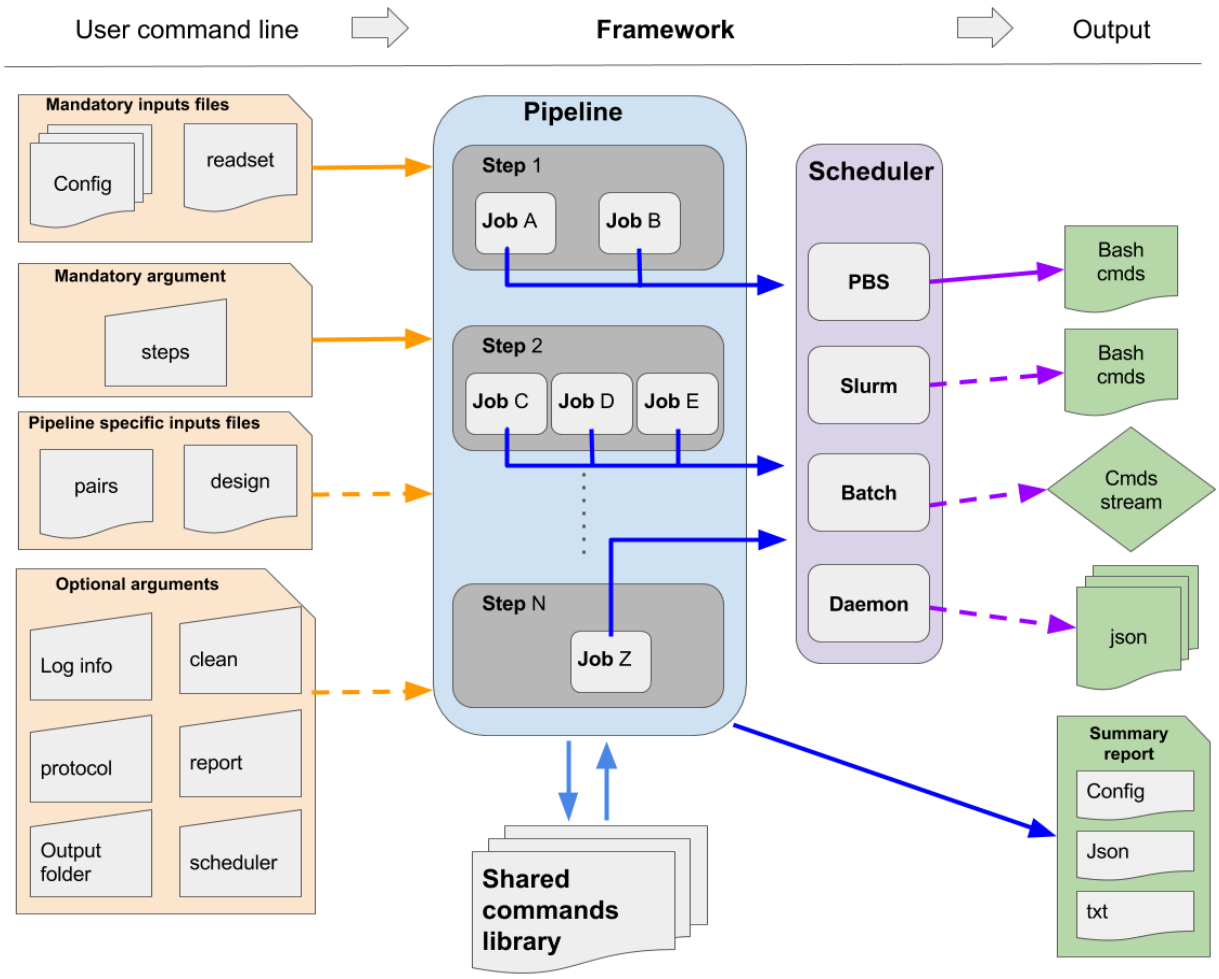
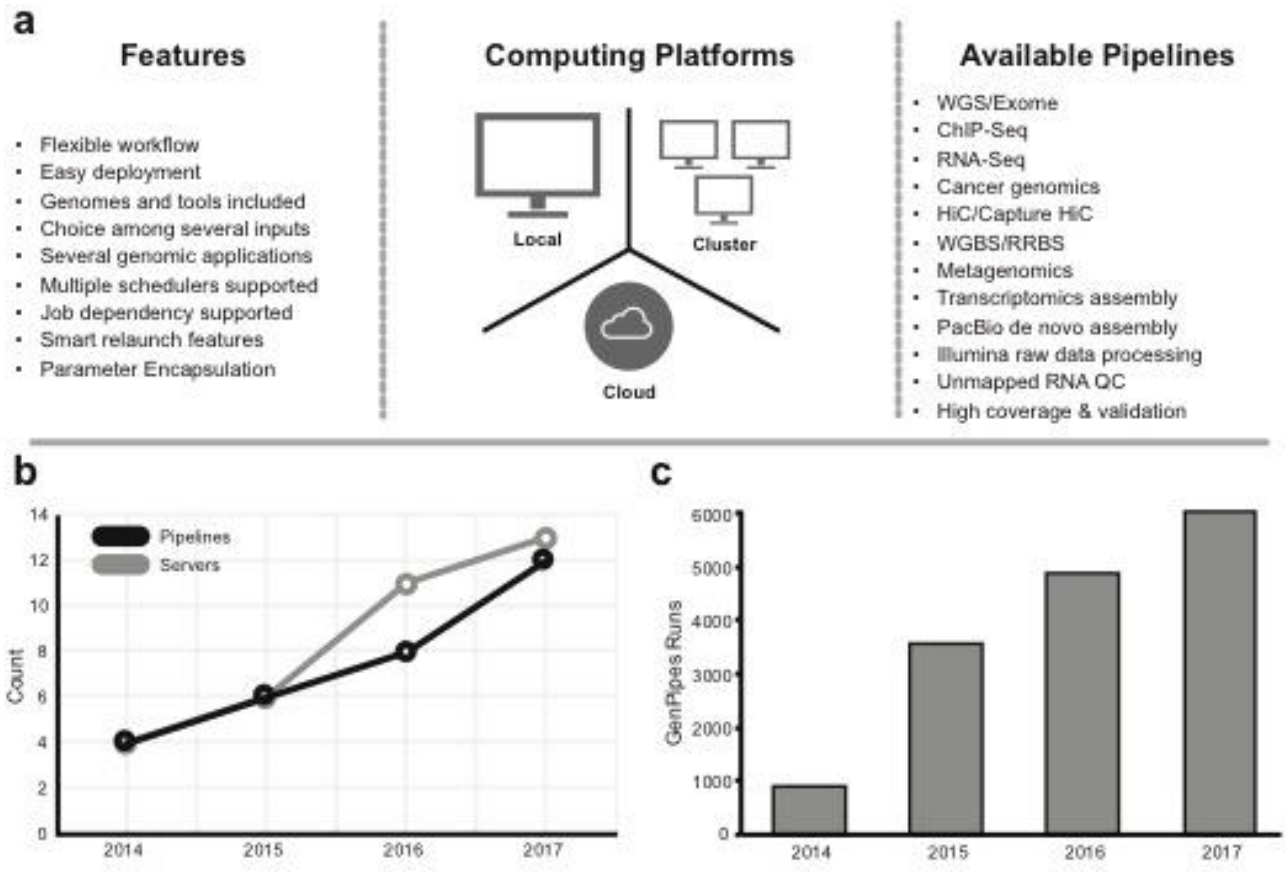


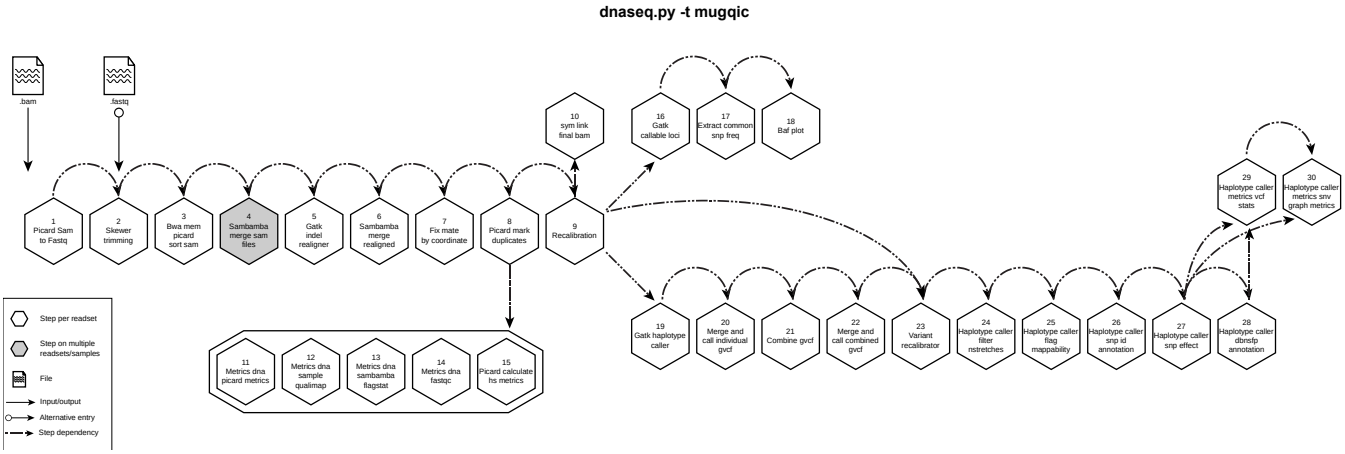
Diagram showing how the information flows from the user command line input through the 4 different objects (*Pipeline, Step, Job and Scheduler*) in order to generate system specific executable outputs.

Figure 2 - **GenPipes properties**



GenPipes' properties and growth. (a) Diagram showing GenPipes' features, compatible computing platforms and available pipelines. (b) GenPipes' available pipelines and maintained servers since the release of GenPipes in 2014. (c) Bar plot showing the number of GenPipes runs per year since its release.

Figure 3 – GenPipes DNASeq pipeline diagram



Schematic representation of GenPipes' dnaseq.py pipeline. Hexagons represent steps in the pipeline. White hexagons represent steps that process input from a single sample, while grey ones represent steps that process input from several samples. Arrows show step dependencies.

Table 1 - Comparison of available solutions for NGS analysis.

	Features												Pipelines								
Solution	Language	Software license	Published	Free	Open source	Cloud/Container	HPC	Workflow manager	Progress Monitoring	GUI	Reports	Config Validation	Germline	Somatic	RNA-Seq	RNA-Seq De novo	ChIP-seq	Metagenome	Methyl-Seq	Hi-C	PacBio assembly
GenPipes	Python	GNU LGPL	Pending	✔	✔	✔	✔	✔	✔	✘	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Genome Modeling System	Perl	GNU LGPLv3	[57]	✔	✔	✔	✔	✔	✔	✘	✔	✔	✔	✔	✔	✘	✘	✘	✘	✘	✘
Galaxy	Python	Academic Free L3.0	[4]	✔	✔	✔	✔	✔	✔	✔	✘	✘	✔	✔	✔	✔	✔	✔	✘	✘	✘
bcbio-nextgen	Python	MIT License	No	✔	✔	✔	✔	✔	✘	✘	✔	N/A	✔	✔	✔	✘	✔	✘	✘	✘	✘
Omics Pipe	Python	MIT License	[72]	✔	✔	✔	✔	✔	✘	✘	✔	✘	✔	✔	✔	✘	✔	✘	✘	✘	✘
Gene Pattern	Java	Custom	[73]	✔	✔	✔	✔	✔	✔	✔	✔	✘	✔	N/A	✔	✘	✘	✘	✘	✘	✘
Illumina BaseSpace	bash	Custom	No	✘	✘	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✘
BINA Genomic Analysis	Java/Python	Custom	No	✘	✘	✔	✔	✔	✔	N/A	N/A	N/A	✔	✔	✔	N/A	✘	✘	✘	✘	✘
SeqWare	Java	GNU GPLv3	[74]	✔	✔	✔	✔	✔	✔	✔	✔	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘
DNA Nexus Platform	Python/bash	Custom	No	✔	Partial	✔	✘	✔	✔	✔	✔	N/A	✔	✔	✘	✘	✘	✘	✘	✘	✘
gkno	Python	MIT License	No	✔	✔	✔	✔	✔	✘	✘	✘	✔	✔	✘	✘	✘	✘	✘	✘	✘	✘
NGSANE	bash	BSD3	[75]	✔	✔	✔	✔	✔	✔	✘	✔	✘	✔	✘	✘	✘	✘	✘	✘	✘	✘
GATK's Queue	Scala	MIT License & Broad Institute	No	Partial	Partial	✘	N/A	✔	✔	✘	N/A	N/A	✔	✔	✘	✘	✘	✘	✘	✘	✘
CGA's Firehose	Java	N/A	No	✔	✘	N/A	✔	✔	✔	✔	✔	N/A	N/A	✔	✘	✘	✘	✘	✔	✘	✘
MIT STAB	Python	GNU GPLv3	[76]	✔	✔	✔	✘	✔	✘	✔	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘
Cromwell/WDL	Scala	BSD 3-Clause	No	Partial	✔	✔	✔	✔	✔	✘	N/A	✔	✔	✔	✘	✘	✘	✔	✘	✘	✘
BigDataScript	BDS	Apache License V2	[9]	✔	✔	✔	✔	✔	✔	✘	✔	✘	✘	✔	✘	✘	✘	✘	✘	✘	✘
Kronos	Python	MIT license	[77]	✔	✔	✔	✔	✔	✘	✘	✘	✘	✔	✔	✔	✘	✘	✘	✘	✘	✘
Nextflow	Java	GNU GPLv3	[7]	✔	✔	✔	✔	✔	✘	✘	✔	✘	✘	✘	✔	✘	✘	✘	✔	✘	✘
Snakemake	Python	MIT License	[6]	✔	✔	✔	✔	✔	✔	✘	✔	✔	✘	✔	✔	✘	✘	✘	✘	✘	✘

Modified from Griffith & Griffith et al. [57]. Note that community-built pipelines are not considered in the Pipelines section of the table.

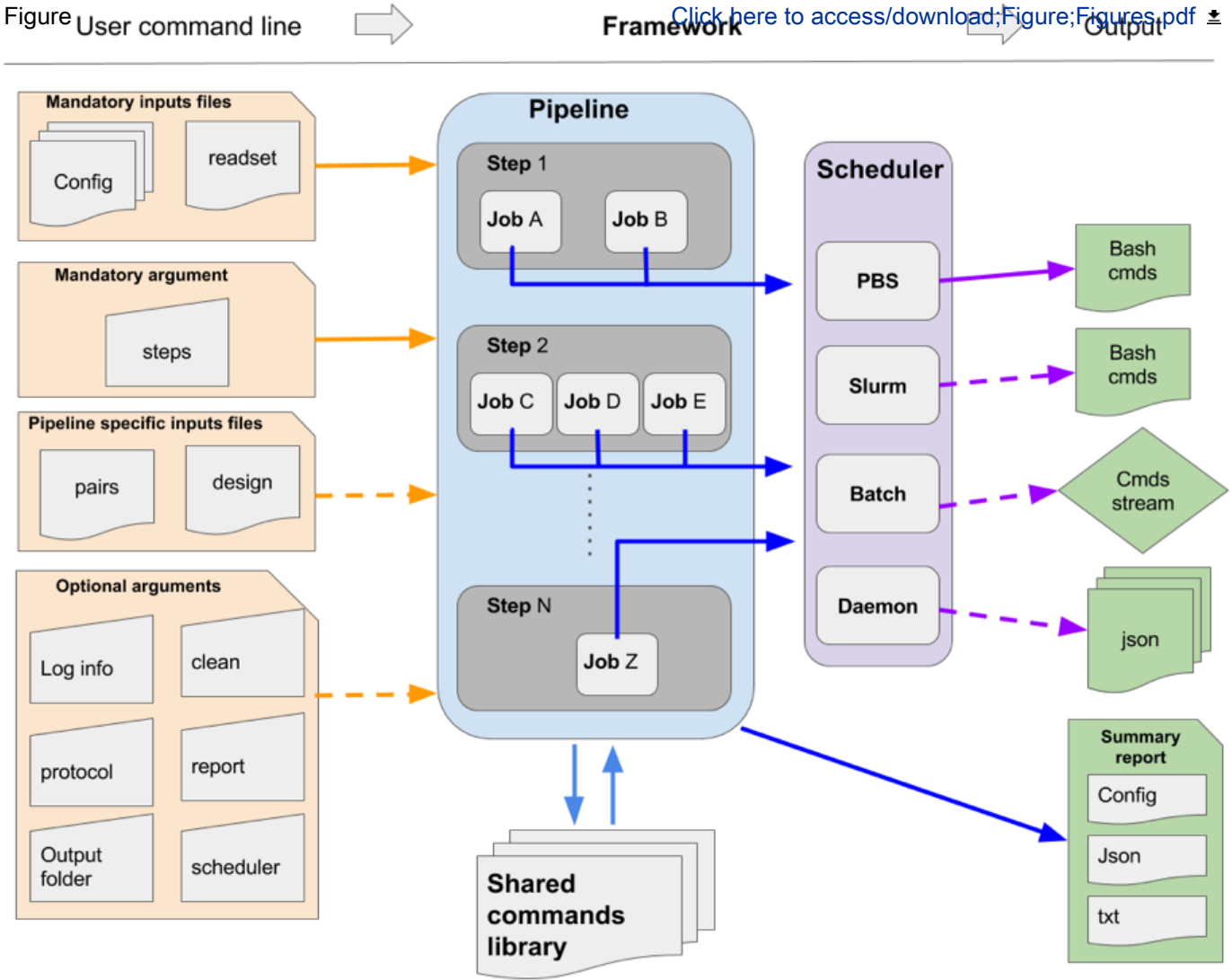
Table

Click here to access/download;Table;Table1_Round2.tiff

	Features												Pipelines								
Solution	Language	Software license	Published	Free	Open source	Cloud/Container	HPC	Workflow manager	Progress Monitoring	GUI	Reports	Config Validation	Germline	Somatic	RNA-Seq	RNA-Seq De novo	ChIP-seq	Metagenome	Methyl-Seq	Hi-C	PacBio assembly
GenPipes	Python	GNU GPL	Pending	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Genome Modeling System	Perl	GNU GPL v3	[97]	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗
Galaxy	Python	Academic Free 3.0	[9]	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✗	✗	✗
bcbio-nextgen	Python	MIT License	No	✓	✓	✓	✓	✓	✗	✗	✓	N/A	✓	✓	✓	✗	✓	✗	✗	✗	✗
Omics Pipe	Python	MIT License	[73]	✓	✓	✓	✓	✓	✗	✗	✓	✗	✓	✓	✓	✗	✓	✗	✗	✗	✗
Gene Pattern	Java	Custom	[73]	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	N/A	✓	✗	✗	✗	✗	✗	✗
Illumina BaseSpace	bash	Custom	No	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗
BWA Genomic Analysis	Java/Python	Custom	No	✗	✗	✓	✓	✓	✓	N/A	N/A	N/A	✓	✓	✓	N/A	✗	✗	✗	✗	✗
SeqWare	Java	GNU GPL v3	[74]	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
DNA Nexus Platform	Python/bash	Custom	No	✓	Partial	✓	✗	✓	✓	✓	✓	N/A	✓	✓	✗	✗	✗	✗	✗	✗	✗
gkno	Python	MIT License	No	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
NGSANE	bash	GNU	[75]	✓	✓	✓	✓	✓	✓	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
GATK's Queue	Scala	MIT License & Broad Institute	No	Partial	Partial	✗	N/A	✓	✓	✗	N/A	N/A	✓	✓	✗	✗	✗	✗	✗	✗	✗
CGA's Firehose	Java	N/A	No	✓	✗	N/A	✓	✓	✓	✓	✓	N/A	N/A	✓	✗	✗	✗	✗	✓	✗	✗
MIT STAR	Python	GNU GPL v3	[76]	✓	✓	✓	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
CopyWeb/WDL	Scala	BSD 3-Clause	No	Partial	✓	✓	✓	✓	✓	✗	N/A	✓	✓	✓	✗	✗	✗	✓	✗	✗	✗
BigDataScript	N/A	Apache License v2	[8]	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗
Kronos	Python	MIT license	[77]	✓	✓	✓	✓	✓	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗	✗	✗	✗
NextFlow	Java	GNU GPL v3	[7]	✓	✓	✓	✓	✓	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗
SnakeMake	Python	MIT License	[8]	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗	✓	✓	✗	✗	✗	✗	✗	✗

Figure

Click here to access/download;Figure;Figures.pdf

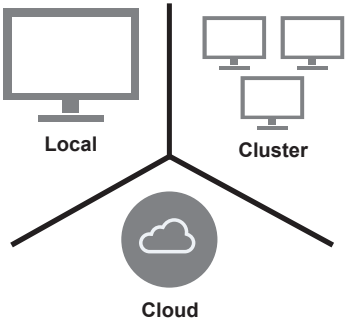


a

Features

- Flexible workflow
- Easy deployment
- Genomes and tools included
- Choice among several inputs
- Several genomic applications
- Multiple schedulers supported
- Job dependency supported
- Smart relaunch features
- Parameter Encapsulation

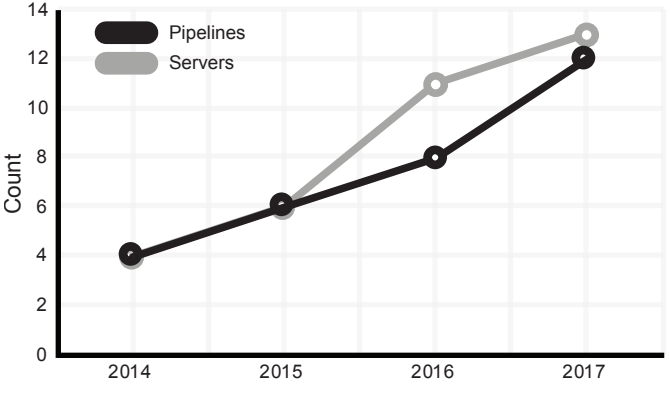
Computing Platforms



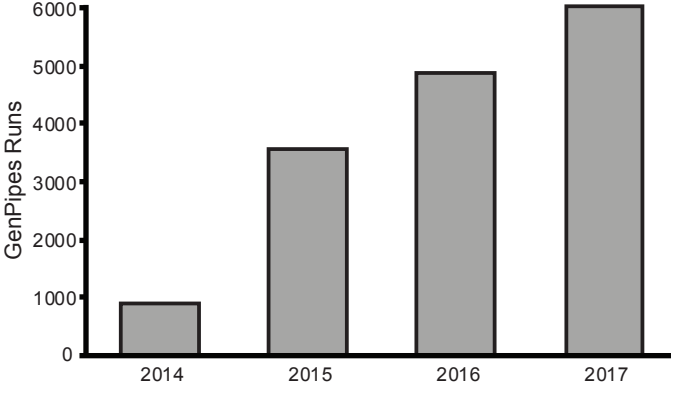
Available Pipelines

- WGS/Exome
- ChIP-Seq
- RNA-Seq
- Cancer genomics
- HiC/Capture HiC
- WGBS/RRBS
- Metagenomics
- Transcriptomics assembly
- PacBio de novo assembly
- Illumina raw data processing
- Unmapped RNA QC
- High coverage & validation

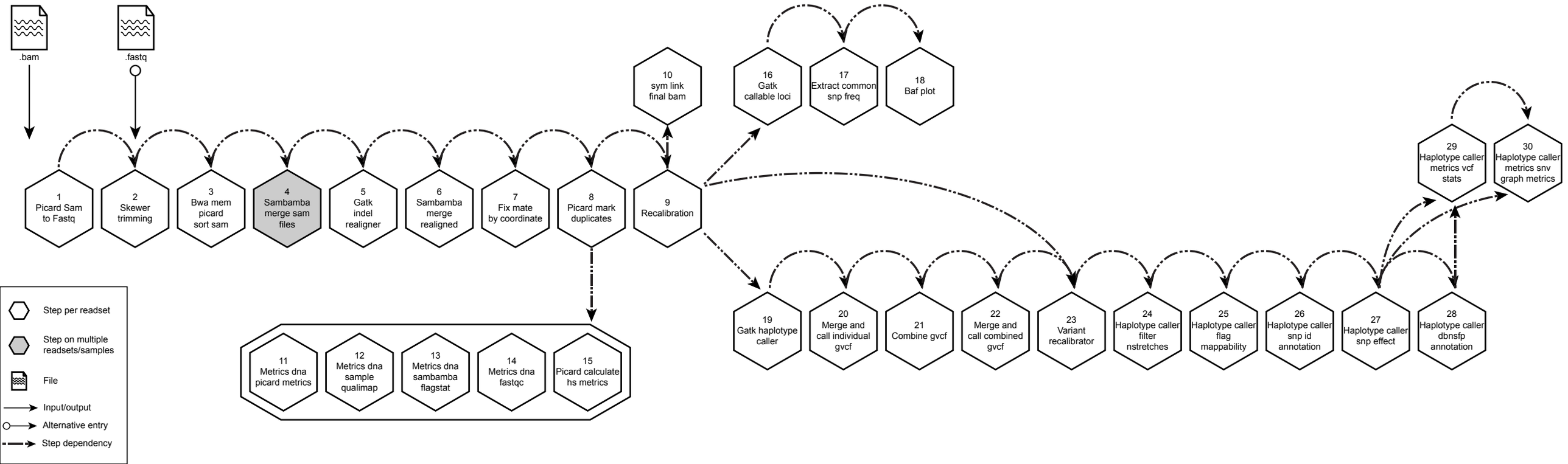
b



c



dnaseq.py -t mugqic



Click here to access/download
Supplementary Material
FigureS1.pdf

Personal Cover

Click here to access/download;Personal Cover;GenPipes_GigaScience_Reviews_Round2_Final.docx

Dear Editor,

Thank you for the opportunity to submit a revised version of the manuscript GIGA-D-18-00198, which addresses the final points raised by the reviewers. Please find our point-by-point response below. New text that has been added to the revised manuscript is shown in red.

Response to the Reviewers:

Reviewer #1: I thank the authors for taking the time to address my previous comments. I believe the manuscript is much stronger as a result and I have no further comments to add.

We thank the reviewer for his constructive criticism that has strengthened the manuscript.

Reviewer #2: The authors successfully address various of *my and my colleagues* requests. However, certain issues remain, which I will list in the following:

Major

* In the introduction, the authors say that frameworks like Galaxy can be inconvenient on large scale projects. Why is that? I think such a claim should be support by a detailed reasoning.

Frameworks like Galaxy are generally web-based. For large scale projects, uploading large datasets to a platform can be time/resource intensive. In general, when projects get larger, it is more efficient to bring the software to the data and not upload the data to the software location.

We have adjusted the text to say:

"… such tools can be inconvenient for large scale projects due to having to move sizeable datasets to the platform".

* When mentioning that WMSs rarely provide pre-built pipelines ready for production analysis, the authors should also mention that they nevertheless support development of such pipelines by the community of users, including linking out to examples like nf-core and github.com/snakemake-workflows.

We have edited the text accordingly:

"It is important to note that GenPipes, as well as several other WMSs, like Nextflow [58] and SnakeMake [59], support community-developed pipelines, however, those have not been included in the comparison."

* In my previous comment, I mentioned that the feature table is biased. While the authors added the columns suggested, these where only meant as examples. I would have thought that the authors take this as incentive to get a less biased view, which is arguably very hard. However, even when only taking the reviewer comments as a base, there are plenty of other columns which should go

into the table. For example, the authors should add "DRMAA support", "status/progress monitoring" as a column. Moreover, the level of cloud support in GenPipes is quite different from what is offered by e.g. Nextflow and Snakemake. There, you have full Kubernetes support, in case of Snakemake even without the requirement of a shared filesystem. Maybe split the cloud column into "basic cloud support" and "kubernetes support".

Our intentions with Table1 was to provide the reader with an overview of the features of several tools in the field but not necessarily an exhaustive list. We did not design the table to be biased towards *GenPipes* as we only modified one of the most comprehensive tables we found in recent manuscripts (Griffith & Griffith et al.). Based on that initial table and reviewers' comments, we added 3 features and 1 WMS. Although more could be added, it would also start cluttering the table and make it difficult to extract meaningful information.

"status/progress monitoring" is already included in the table under "Tracking". We have modified the column name to make it less ambiguous.

Concerning "DRMAA support" and splitting the "cloud support" column into "basic cloud" and "kubernetes support", we feel that this is highly technical/specific for the average user.

* The installation mechanism for new software tools (outside of what is provided out of the box) (explained here: https://bitbucket.org/mugqic/genpipes/src/master/#markdown-header-modules), seems like manually redoing all the work that is already solved by package managers like conda or container engines like singularity. For example, Bioconda provides a library of over 4000 bioinformatics software packages which can be readily used from any WMS that supports conda, and Biocontainers provides the same for container based deployment (which lacks conda's ability to rapidly compose custom combinations of tools though). In order to make the comparison fair, the feature table should therefore contain two columns called "package-manager-integration" and "container-integration". For an example of what level of integration I am referring to, see https://snakemake.readthedocs.io/en/stable/snakefiles/deployment.html#integrated-package-management and https://www.nextflow.io/docs/latest/conda.html?highlight=conda.

Bioconda offers a collection of packages and not an integrated system and can be quite heavy in memory requirements. Hence, we think that "package-manager-integration" is not necessarily an indication of the strength of the WMS. It is a specific choice, one that offers ease of installation but has its pitfalls as well. *GenPipes* does not use package managers by design. GenPipes manages its own libraries making sure there is no conflicting libraries in the process. For users who do not want to install GenPipes manually, we offer a Docker container that has also been tested with Singularity. We have updated the GenPipes' bitbucket documentation to highlight the availability of the GenPipes' Docker container.

"container-integration" has already been included in the table under the "Cloud/Container" column.

* I am pleased to see that GenPipes indeed supports aggregation over many samples. What remains is the question whether the only entity to aggregate over are samples. If so, only over all samples or is it possible to express e.g. an arbitrary grouping of samples? Moreover, what about other properties, e.g. for scanning a parameter space? I suggest to somehow reflect the different ways of aggregation in the feature table, maybe using the terms that I mentioned in my first review.

GenPipes is a flexible python framework that aggregates over readsets, samples and other entities, like chromosomes, based on the pipeline. Arbitrary groupings of samples can be defined in pipelines that use design files, like chipseq and rnaseq. Scanning parameter space can be done by adjusting the configuration files. There isn't a set of limited/defined aggregation methods we use; aggregation is used based on each pipeline's needs. The user can refer to the documentation of each pipeline to see what is possible. For user implemented pipelines, there is no restriction on the aggregations possible.

We have added the following lines to the text to highlight some of these points:

"... GenPipes can aggregate and merge samples as indicated by the readset file."
"... Configuration files are customizable, allowing users to adjust different parameters."
"... Custom sample groupings can be defined in the design file."

Minor

* Please mention in the caption of the feature table that community based workflows are not considered in the comparison. It might otherwise be that readers overlook this in the main text.

We have added test to the caption as follows:

"Modified from Griffith & Griffith et al. [57]. Note that community-built pipelines are not considered in the Pipelines section of the table."

* Figure S1 contains a lot of typos, e.g. "reasdet", which I guess is supposed to be readset?

Thank you, we have corrected the typos in Figure S1.