

Manuscript Number:	GIGA-D-18-00198R1	
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:	<p>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.</p>	
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	

	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:	<p>Dear GigaScience Editorial Office,</p> <p>Thank you for opportunity to submit a revised version of the manuscript GIGA-D-18-00198 that addresses the points raised by the two reviewers. Following the constructive comments and suggestions, we have made several major improvements to the manuscript, including:</p> <ul style="list-style-type: none"> -The reviewers had comments about the dependencies between steps, as well as whether steps worked on single samples or a cohort of samples. To help answer these questions, we have added a diagram summarizing each pipeline to the supplementary. -We have added extensive benchmark metrics for two pipelines, dnaseq and tumour_pair. -We have added documentation on how to run GenPipes on a cloud platform on our website and now referenced it in the manuscript. -GenPipes is now registered in the SciCrunch database under the RRID SCR_016376. The RRID has been included in the revised manuscript under the 'Availability and requirements' section. -We have re-organized the text a bit to better explain the unique features of GenPipes <p>See below for our point-by-point response to the reviewers. New text that has been added is shown in red in the revised manuscript.</p> <p>Response to the Reviewers:</p> <p>Reviewer #1: The authors present in this manuscript both a new workflow management system (GenPipe), as well as a set of bioinformatics pipelines that are built to run on this system. The authors contribution is likely be of interest to many genome centres and bioinformaticians, who wish to leverage existing pre-built and tested pipelines. The manuscript is clear and well written and the source code is well structured, extensive, and is well documented. The developers have also taken steps to ease installation and configuration issues that might occur when trying to install the software in other environments.</p> <p>However, I have reservations regarding the structure and content of the manuscript. I find it lacks detail and analysis that would convince a reader to adopt their system, both with regards to GenPipe itself, as well as the pipelines. This is unfortunate as I think the authors have provided a large contribution to the field in making available their resources.</p> <p>We thank the reviewer for evaluating the manuscript and for his positive view of our contribution to the bioinformatics community. We have now re-organized the text a bit</p>

to better explain the unique features of GenPipes and have added more information to the manuscript that should provide the users with a better overview of its advantages (see below).

Major points:

1. The authors only provide a superficial comparison to existing systems. A more detailed analysis of why new pipeline developers should use GenPipe over an alternatives? What distinguishes this as a WMS from SnakeMake for example? From what I can see in the manuscript there are several implementation details within GenPipe that appear sub-optimal, which I'll elaborate on in the points below.

In our original submission, we attempted to provide a comparison of GenPipes to other available WMSs. While far from exhaustive, we had compared GenPipes to 18 popular WMSs by looking at 9 different features and 9 different pipelines (Table 1). In the revised submission, we have added SnakeMake to the WMS list and included 3 more features suggested by reviewer #2 (see below). We have also added a detailed workflow for each pipeline (Figure 3 and S1-14) and have added new benchmarking metrics for some of the pipelines. We also included estimate resource usage for each pipeline across different servers (Table S1). We have also moved the description of the unique features of GenPipes to the Result section of the manuscript. We think that this new information will hopefully give potential users a better idea of how GenPipes compares to some of the existing frameworks available and its advantages.

2. I would also like to see a proper analysis for each pipeline (can be provided in supplemental information) describing comparisons to existing pipelines in terms of accuracy, resource usage, runtime stats, etc.

We agree with the reviewer that benchmarking the individual pipelines for accuracy, resource usage and runtime statistics is important. Along those lines, we have added benchmarks for several pipelines to the Supplementary Material. In terms of resource usage and runtime statistics, it has been our experience that these metrics vary widely depending on system hardware, software versions and sample sequencing depth. However, we agree that a ballpark estimate of these resources would be useful for the user and have added Table S1.

In terms of accuracy, it is important to remember that GenPipes is a framework that is built around open source, third party tools that are available to the scientific community. Accuracy is not as easily assessed in certain fields due to the lack of a good quality "truth" set to benchmark against. Generally, we have tried to model GenPipes pipelines following large scale projects like GATK best practices SOPs and ENCODE and have relied on public benchmarking, in addition to our own.

Minor points:

1. Introduction: "Such solutions are flexible and can help in pipeline implementation but do not provide robust standardized pipelines which are ready for production-scale analysis." In my experience, it's simply not true that WMS solutions are not suitable for production scale analysis. There are many examples of people doing exactly this, and moreover I've built several myself which are run multiple times every day without issue.

In my experience they can work very reliably, and ability to tolerate and resume from errors is easy to code in. It is also unclear what the authors mean by standardisation in this context. I'd request that the authors either justify this point and provide concrete examples of exactly what the source of the perceived issues are, or remove this sentence.

We agree with the reviewer that the sentence is not delivering the idea we intended. The sentence has been edited in the manuscript to highlight the fact that not all WMSs come with pre-built pipelines that are ready for use. We agree with the reviewer that many WMSs are robust and powerful. We are simply trying to appeal to both the advanced user with GenPipes's WMS and the novice user with the pre-built and tested pipelines.

The sentence now reads:

"Such solutions are flexible and can help in pipeline implementation but rarely provide robust pre-built pipelines which are ready for production analysis."

2. Introduction: "These are useful for specific applications but can be challenging to implement, difficult to modify or scale-up. They have also rarely been tested on multiple

computing infrastructures." This seems too strong a statement. In some cases this might be true but there are many examples of robust pipelines that efficiently leverage data centre hardware.

We agree with the reviewer that the statement might be generalized beyond context. We have edited the text to explicitly prevent the statement from applying to all pipelines.

The sentence now reads:

"These are useful for specific applications but can sometimes be challenging to implement, difficult to modify or scale-up."

3. Introduction: "GenPipes has been tested, benchmarked ...". It is not clear whether the "testing and benchmarking" refers to the pipelines or the WMS itself. This should be clarified.

Both GenPipes's pipelines and the WMS have been tested extensively. The pipelines have been benchmarked and have been used to process thousands of samples. The WMS has been stress-tested and adapted to different computing infrastructure and is currently run on at least 6 super computers that we help maintain. We have modified the text to clarify this.

The sentence now reads:

"GenPipes' WMS and pipelines have been tested, benchmarked and used extensively over the past four years."

4. Schedulers: Ideally, GenPipes should offer the ability to implement scheduling via DRMAA which would increase the potential sites that could potentially run genpipe.

For example, currently any data centres running Platform LSF could not use genpipe but via4 DRMAA this would be possible.

We thank the reviewer for his suggestion. We generally like to minimize the number of layers between GenPipes and the system it is running on, and did not consider DRMAA previously. We have added the DRMAA scheduler to our potential future projects and have mentioned it in the discussion.

5. Job dependencies: I have reservations that the approach taken here is optimal. If I understand correctly, job dependencies are setup using the selected scheduler and all jobs, across steps, are launched at the same time. I suspect for very large pipelines containing many thousands jobs (not uncommon) this would put an undue burden on the scheduler and therefore would not scale very well. Could the authors elaborate on this point and highlight details such as what happens when a pipeline fails? Are existing jobs explicitly terminated? Or somehow left running and continue after the pipeline is resumed?

We agree with the reviewer that this feature may be optimized and have been working on this for a future GenPipes release. It is not ready yet, as we would like to optimize and test all supported schedulers before releasing the new feature. The current process works by creating the full script which is communicated to the scheduler. This approach has been initially chosen for its low level of complexity and reproducibility; the script is readable and editable by a minimally trained user and can also be re-run later on. This also avoid having to generate local processes which would need to stay in active mode in order to monitor the job submission process. In order to help monitoring the pipeline progress, GenPipes includes a script (logReport.pl) that generates a report of the current status of the pipeline. We have also included a JSON log file system that could be used to develop a local web-portal displaying the pipeline job status in real time.

We have been using GenPipes on hundreds of samples every day, submitting thousands of jobs, and has not run into any issues with our compute providers so far. However, we do intend to optimize the process by using job arrays.

In terms of what happens when the pipeline fails, we have added a section to the text under "Running GenPipes" to elaborate on this point:

"... 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..."

6. Configuration Files: "Configuration files, also referred to as "ini" files, are provided among the arguments of the GenPipes command.". The authors should change the wording here. "Ini" is a legacy windows-based configuration file format. I'm not asking for the authors to change the configuration format used but it would be useful to have some justification for this unusual choice. Alternatives, like "yaml" for example allow for stricter and richer structuring and is therefore much easier to parse and in turn normally results in less buggy code.

We agree with the reviewer that many file configuration formats exist and that some might be considered more optimal than the legacy ini file format. However, we chose the ini format for its readability and the ease of use of the ini schema (section, key and value). It can be edited manually with any text editor without the need to worry about syntax or indentation which is easier for standard analysts using GenPipes for their analysis. Additionally, the python language offers a standard library (configParser) that is made to easily integrate this standard configuration format.

Reviewer #2: The manuscript presents GenPipes, a Python-based framework for defining and executing data analysis workflows.

GenPipes is based on a handful of Python classes that can be inherited and implemented to achieve a formal and executable description of a workflow in terms of steps. During execution, steps are specialized to jobs that perform concrete operations on input files.

For me, the most important, and definitely valuable addition of this work is the comprehensive collection of well-tested workflows covering the most important applications of sequencing.

In general, I think this should be emphasized more, at the expense of removing some of the weaker aspects of the paper. I will outline this below.

Major Comments

* The manuscript argues that a major advantage of GenPipes is the rich collection of production-ready workflows that are delivered with the system. The list of workflows is indeed impressive, it should be mentioned though that both Snakemake and Nextflow also provide (community-maintained) collections of tested workflows, like github.com/snakemake-workflows, nf-core.github.io and [sequana](https://sequana.org). I agree though that it might very well be that these are still less mature (except sequana), as they are probably newer.

We thank the reviewer for seeing value in our growing collection of pipelines. In our manuscript, we avoided reference to community-maintained workflows. While community-maintained workflows are a great testament to the usefulness of a WMS, they are hard to keep track of and evaluate. GenPipes supports community-maintained workflows as well, however, those too have not been mentioned in the manuscript. The pipelines that have been mentioned are pipelines that have been validated and are maintained by the tool authors, which we think is an important distinction.

We have made this clear in the text, as follows:

"It is important to note that GenPipes, as well as several other WMSs, have community-supported pipelines, however, those have not been included in the comparison."

* The manuscript claims that GenPipes supports cloud execution, but I cannot find a scheduler for this purpose in the list of schedulers on page 4. Also, the feature table says that cloud support is pending.

Yes, GenPipes supports cloud execution via a container image and not a particular scheduler. Through the container image, any available scheduler can be used, depending on the cloud architecture in place. We apologize for the omission in Table 1, we have fixed it. We have now also documented the use of GenPipes in the cloud and added a user manual at:

<http://www.computationalgenomics.ca/genpipes-in-the-cloud/>

* On page 7, when describing deployment of software and reference information, it is

unclear whether installation happens system wide (needing admin rights) or local. This should be clearly stated, since system-wide installation would be a major disadvantage compared to systems like Nextflow, Snakemake or CWL based WMSs. Moreover, it should be mentioned how those dependencies are updated, and in what sense such updates would affect previous runs, which could potentially lose reproducibility, if updates happen globally.

GenPipes installation can happen both system-wide or locally as the pipeline will only use software and modules provided in a specific path defined through an environment variable. All the third part tool installation scripts provided with the pipeline have been designed to work on a local path system and do not require root privileges. We also developed a container image of GenPipes which runs GenPipes with little software installation. This allows larger processing centers to install GenPipes for all users, but also allows individuals the flexibility to adopt GenPipes for their own needs without needing special permissions or setup.

We have added text in the result section to explain these points and to expand on the dependency updates:

“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.”

* In the discussion, it is mentioned that GenPipes is currently being reimplemented in WDL. It is a good choice to use one of the established, more feature-rich systems. However, then, large parts of this paper are in fact obsolete, as they will be replaced with WDL. The major contribution that remains after that step is the collection of workflows, which is totally fine, since this is a very valuable addition. I therefore suggest to put more focus on the workflows, and simply outline that they are currently implemented in GenPipes and soon will be available in WDL. Moreover, choice of tools, parameters and how the benchmarking was done (in a more concrete way instead of simply saying "we used GIAB") should be described in detail.

Actually, based on recent developments, we are probably going to go with CWL over WDL. That being said, we do not fully agree with the reviewer that this will make GenPipes WMS obsolete. While CWL or WDL can help increase the compatibility of GenPipes with different systems, it will add a layer of complexity to GenPipes; one that is not needed on HPC systems. We agree with the reviewer that the pipelines are a major contribution of GenPipes and that they deserve a more detailed description. We have now added more descriptions and benchmarks as supplementary material. We have also added workflow diagrams representing pipeline workflow and dependencies in the supplementary.

* Table 1 provides a feature comparison. As with every single feature comparison I have seen so far, it is highly biased, showing only features that GenPipes itself provides. For example, GUI (as provided e.g. by Galaxy) and automatic reports are missing. Per-step/job software deployment and container support is missing. Config file validation is missing. Items are not sufficiently explained (e.g., what is meant with tracking, and in what sense is Nextflow not providing it). A popular system is completely missing from the table: Snakemake. Via nf-core and other projects, Nextflow and Snakemake provide several of the mentioned pipelines. Finally, I cannot actually find that table in reference [62], although the authors claim that it is a modified version of the table from that paper.

We thank the reviewer for his excellent suggestions. We have added 3 more features to the comparison table (GUI, Reports and Config validation). We have also added SnakeMake to the comparison. In evaluating the pipelines available, we have looked at the published manuscript if it exists or the code base and documentation of each tool. Community-maintained pipelines were not considered, as it becomes difficult to draw the line on what to include in the comparison and it is hard to assess the reliability of these pipelines without extensive benchmarking. Only pipelines provided by the tool authors were considered.

	<p>Finally, the comparison Table1 was modified from Griffith et al. in supplementary material (Table S6) which contains a simple version of the table. following the link in the figure leads to the full version of the table:</p> <p>https://journals.plos.org/ploscompbiol/article/file?id=info%3Adoi/10.1371/journal.pcbi.1004274.s021&type=supplementary</p> <p># Minor Comments</p> <p>* On page 2, when mentioning other WMSs, the authors should also mention Nextflow. Moreover, CWL and WDL are not WMSs, and should be listed separately as "declarative workflow description languages".</p> <p>We have edited the text as suggested.</p> <p>* On page 5, when relaunch features are mentioned, common functions of other systems like manual forcing or handling of missing files are not mentioned. Are these not available?</p> <p>GenPipes supports manual forcing through the "-f" option. GenPipes validates the existence of all required modules and genome files and input files in the config file before creating the commands. If any are missing, it looks for alternative files using the ordered list of input implementation described in the "Key GenPipes features options". If none of the possible files are found, GenPipes will throw a Missing File exception and terminate. The text has been edited to clarify this.</p> <p>* On page 6, the description of input choice does not really make it clear how multiple input files or aggregation is handled. It would be beneficial to see examples for (a) a 1-in-1-out job, (b) an aggregating job, (c) a scattering job, (d) a mixed job (n-in-m-out).</p> <p>GenPipes has an array of steps with different behaviors. Some steps operate on a single sample input while others operate on the cohort of available samples (metric steps). To try to distinguish these, we added color coding (back/white) to the workflow diagrams we added in supplementary Figure 1. The dependencies between steps are mapped by GenPipes through input and output files required and communicated to the scheduler which then coordinated job launch. For a full list of pipelines and steps, please refer to Figure S1-14.</p> <p>* On page 8: "all workflows accepts a bam or fastq file as input". I guess they accept multiple bams or fastqs, right? Otherwise they could only be applied to a single sample at a time...</p> <p>To take full advantage of HPC power and reduce processing times, GenPipes runs each sample separately, when possible. Some steps aggregate inputs from many samples at a time. Those have been colored in black in supplementary Figure2 S1-14.</p>
Additional Information:	
Question	Response
Are you submitting this manuscript to a special series or article collection?	No
<p>Experimental design and statistics</p> <p>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.</p> <p>Have you included all the information requested in your manuscript?</p>	Yes

<p>Resources</p> <p>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.</p> <p>Have you included the information requested as detailed in our Minimum Standards Reporting Checklist?</p>	<p>Yes</p>
<p>Availability of data and materials</p> <p>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.</p> <p>Have you have met the above requirement as detailed in our Minimum Standards Reporting Checklist?</p>	<p>Yes</p>

[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 Fillon⁵, Marc Michaud², Maxime Caron^{1,2,5}, Johanna Sandoval⁶, Francois Lefebvre^{1,2}, Gary Leveque^{1,2}, Eloi Mercier^{1,2}, David Bujold^{1,2}, Pascale Marquis^{1,2}, Patrick Tran Van⁷, David Morais⁸, Julien Tremblay⁹, 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. **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

1
2
3
4 platforms, including cloud platforms. This allows system-wide installations, as well as local user installations
5 via the Docker image without needing special permissions.
6
7

8 Through a partnership with the Compute Canada consortium (<https://www.computecanada.ca>), the
9 pipelines and third-party tools have also been configured on 6 different Compute Canada HPC centers. It
10 allows any Canadian researcher to use GenPipes along with the needed computing resources by simply
11 applying to the consortium [10]. To ensure consistency of pipeline versions and used dependencies (such
12 as genome references and annotation files) and to avoid discrepancy between compute sites, pipeline
13 setup has been centralized to one location which is then distributed on a real-time shared file system: the
14 CERN Virtual Machine File System [11].
15
16
17
18
19
20
21

22 **Running GenPipes**

23
24 GenPipes is a command line tool. Its use has been simplified to accommodate general users. A
25 full tutorial is available [12]. Briefly, to launch GenPipes, the following is needed:
26
27

- 28 • A readset file that contains information about the samples, indicated using the flag “-r”.
 - 29 • Configuration/ini files that contain parameters related to the cluster and the third-party tools,
30 indicated using the flag “-c”.
 - 31 • The specific steps to be executed, indicated by the flag “-s”.
- 32
33
34

35 The generic command to run GenPipes is:
36

```
37 <pipeline>.py -c myConfigurationFile -r myReadSetFile -s 1-X > Commands.txt && bash Commands.txt  
38  
39
```

40 Where <pipeline> can be any of the 12 available pipelines and X is the step number desired. Commands.txt
41 contains the commands that the system will execute.
42
43

44 Pipelines that conduct sample comparisons, like ChIP-Seq and RNA-Seq, require a design file that
45 describes each contrast. Design files are indicated by the flag “-d”. The tumour_pair pipeline requires
46 normal-tumour pairing information provided in a standard CSV file using the “-p” option. For more
47 information on the design file and the content of each file type, please consult the GenPipes tutorial and
48 the online documentation.
49
50
51

52 When the GenPipes command is launched, required modules and files will be searched for and
53 validated. If all required modules and files are found, the analysis commands will be produced. GenPipes
54 will create a directed acyclic graph (DAG) that defines job dependency based on input and output of each
55 step. For a representation of the DAG of each pipeline, refer to supplementary figures S1-14. Once
56 launched, the jobs are sent to the scheduler and queued. As jobs complete successfully, their dependent
57
58
59
60
61
62
63
64
65

1
2
3
4 jobs are released by the scheduler to run. If a job fails, all its dependent jobs are terminated and an email
5 notification is sent to the user. When GenPipes is re-run, it will detect which steps have successfully
6 completed, as described in section ‘Smart relaunch features’, and skip them but will create the command
7 script for the jobs that were not completed successfully. To force the entire command generation, despite
8 successful completion, the “-f” option should be added.
9
10
11
12
13

14 15 **RESULTS**

16
17 GenPipes was first released in 2014. Since then, it has grown to implement 12 pipelines and is
18 currently installed and maintained on 13 different clusters (Figure 2a-b). GenPipes has been actively used
19 for the last four years to quality control and analyze thousands of samples each year (Figure 2c). It has also
20 been used to analyze data for several large-scale projects such as IHEC [2] and eFORGE [13].
21
22
23
24
25

26 27 **Key features of GenPipes**

28
29 GenPipes’ framework has been optimized to facilitate large scale data analysis. Several features
30 make this possible (Figure 2a):
31
32

33 34 **Multiple schedulers**

35
36 GenPipes is optimized for HPC processing. It can currently accommodate four different types of
37 schedulers:
38

- 39 ● *PBSScheduler* creates a batch script that is compatible with a PBS (TORQUE) system.
- 40 ● *SLURMScheduler* creates a batch script that is compatible with a SLURM system.
- 41 ● *BatchScheduler* creates a batch script which contains all the instructions to run all the jobs one
42 after the other.
- 43 ● *DaemonScheduler* creates a log of the pipeline command in a JSON file.
44
45
46
47

48 49 **Job dependencies**

50
51 In order to minimize the overall analysis time, GenPipes uses a dependency model based on input
52 files, which is managed at the *Job* object level. A job does not need to wait for the completion of a previous
53 step unless it is dependent on its output. Jobs thus become active and can be executed as soon as all their
54 dependencies are met, regardless of the status of previous jobs or of other samples. Thus, when a pipeline
55 is run on multiple samples, it creates several dependency paths, one per sample, each of which completes
56 at its own pace.
57
58
59
60
61
62
63
64
65

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

1
2
3
4 order in the list. The first input file found either on disk or in the overall output list will be chosen as input.
5 The chosen input will determine the dependency of the *Job* to the other *Jobs* in the pipeline. This system
6 is really flexible and allows users to skip specific steps in the pipeline if they consider them unnecessary.
7
8

9 **Customizable workflows**

10
11 Despite the benchmarking and testing made on the standard analysis procedures implemented in
12 GenPipes, some users may be interested in modifying pipelines. In order to make GenPipes more flexible,
13 a *protocol* system is used. The system allows the implementation of different workflows into a single *Pipeline*
14 object. As a result, one can replace specific steps by other user specific ones. In that case, the user will
15 only need to implement these new Steps and define an additional protocol which will use part of the initial
16 Steps and the newly developed ones. As an example, this has been used to incorporate the Hi-C analysis
17 workflow and the capture Hi-C analysis workflow into GenPipes' hicseq pipeline. A flag (-t hic or -t capture)
18 can be used to specify the workflow to be executed. This system has been developed to reduce the amount
19 of work for external users that decide to contribute to code development and to limit the number of Pipeline
20 objects to maintain. **This will also allow us to provide multiple workflows per pipeline to appeal to different
21 tool preferences in each field.**
22
23
24
25
26
27
28

29 **Facilitating dependency installation**

30
31
32 Genomic analyses require third party tools, as well as genome sequence files, annotation files and
33 indices. GenPipes comes configured with a large set of reference genomes and their respective annotation
34 files, as well as indices for most aligners. It also includes a large set of third party tools. If GenPipes is being
35 installed from scratch on new clusters, automatic bash scripts that download all tools and genomes are
36 included to ease the setup process. **These scripts support local installations without the need for super-
37 user privileges. Tools and dependencies are versioned and are loaded by GenPipes in a version-specific
38 manner. This allows different pipelines to use different software versions based on need. It also allows
39 retention of the same parameters and tools for any given project for reproducibility. GenPipes is also
40 provided as a container version for which no dependency installation is required.**
41
42
43
44
45
46
47
48

49 **Available workflows**

50
51 GenPipes implements 12 standardized genomics workflows including: DNA-Seq, Tumour Analysis,
52 RNA-Seq, de novo RNA-Seq, ChIP-Seq, PacBio assembly, Methyl-Seq, Hi-C, capture Hi-C, and
53 Metagenomics (Figure 2c). All pipelines have been implemented following a robust design and development
54 routine by following established gold standards standard operating protocols (SOP). Below we summarize
55 GenPipes' workflows; more details are available in the GenPipes documentation. **For more details**
56
57
58
59
60
61
62
63
64
65

1
2
3
4 concerning computational resources used by each pipeline, refer to supplementary Table S1. All workflows
5 accept a bam or a fastq file as input.
6

7 DNA-Seq Pipeline: 8 9

10 DNA-Seq has been implemented optimizing the GATK best practices SOPs [14]. This procedure
11 entails trimming raw reads derived from whole genome or exome data followed by alignment to a known
12 reference, post alignment refinements and variant calling. Trimmed reads are aligned to a reference by the
13 Burrows-Wheeler Aligner, bwa-mem [15]. Refinements of mismatches near indels and base qualities are
14 performed using GATK indels realignment and base recalibration [14] to improve read quality post
15 alignment. Processed reads are marked as fragment duplicates using picard mark duplicates [14] and SNP
16 and small indels are identified using either GATK haplotype callers or samtools mpileup [16]. The Genome
17 in a Bottle [17] dataset was used to select steps and parameters minimizing the false positive rate and
18 maximizing the true positive variants to achieve a sensitivity of 99.7%, precision of 99.1% and F1-score of
19 99.4% (For more details, refer to Supplementary Materials). Finally, additional annotations are incorporated
20 using dbNSFP [18] and/or Gemini [19XX] and quality control metrics are collected at various stages and
21 visualized using MultQC [20]. This pipeline has two different protocols, the default protocol based on the
22 GATK variant caller, haplotype_caller, (“-t mugqic”, Figure 3) and one based on the mpileup/bcftools caller
23 (“-t mpileup”, Figure S1). Another pipeline that is optimized for deep coverage samples,
24 dnaseq_high_coverage, can be found in Figure S2.
25
26
27
28
29
30
31
32
33

34 RNA-Seq Pipeline: 35 36

37 This pipeline aligns reads with STAR [21] 2-passes mode, assembles transcripts with Cufflinks [22]
38 and performs differential expression with Cuffdiff [23]. In parallel, gene-level expression is quantified using
39 htseq-count [24], which produces raw read counts that are subsequently used for differential gene
40 expression with both DESeq [25] and edgeR [26]. Several common quality metrics (rRNA content,
41 expression saturation estimation etc.) are also calculated through the use of RNA-SeQC [27] and in-house
42 scripts. Gene Ontology terms are also tested for over-representation using GOseq [28]. Expressed short
43 SNVs and indels calling is also performed by this pipeline, which optimizes GATK best practices to reach
44 a sensitivity 92.8%, precision 87.7% and F1-score 90.1%. A schema of pipeline steps can be found in
45 Figure S3. Another pipeline, maseq_light, based on Kallisto [29] and used for quick quality control can be
46 found in Figure S4.
47
48
49
50
51
52

53 De-Novo RNASeq Pipeline: 54 55

56 This pipeline is adapted from the Trinity-Trinotate suggested workflow [30] [31]. It reconstructs
57 transcripts from short reads, predicts proteins and annotates leveraging several databases. Quantification
58 is computed using RSEM and differential expression is tested in a manner identical to the RNA-seq pipeline.
59 We observed that the default parameters of the Trinity suite are very conservative which could result in the
60
61
62
63
64
65

1
2
3
4 loss of low-expressed but biologically relevant transcripts. In order to provide the most complete set of
5 transcripts, the pipeline was designed with lower stringency during the assembly step in order to produce
6 every possible transcript and not miss low expressed mRNA. A stringent filtration step is included afterward
7 in order to provide a set of transcripts that make sense biologically. A schema of pipeline steps can be
8 found in Figure S5.
9

10 ChIP-Seq Pipeline:

11
12 The ChIP-Seq workflow is based on the ENCODE [1] workflow. It aligns reads using the Burrows-
13 Wheeler Aligner. It creates tag directories using Homer [32]. Peaks are called using MACS2 [33] and
14 annotated using Homer. Binding motifs are also identified using Homer. Metrics are calculated based on
15 IHEC requirements [34]. The ChIP-Seq pipeline can also be used for ATAC-Seq samples. However, we
16 are developing a pipeline that is specific to ATAC-Seq. A schema of pipeline steps can be found in Figure
17 S6.
18
19

20 The Tumour Analysis Pipeline:

21
22 The Tumour Pair workflow inherits the bam processing protocol from DNA-seq implementation to
23 retain the benchmarking optimizations but differs in alignment refinement and mutation identification by
24 maximizing the information utilizing both tumour and normal samples together. The pipeline is based on an
25 ensemble approach, which was optimized using both the DREAM3 challenge [35] and the CEPH mixture
26 datasets to select the best combination of callers for both SNV and SV detection. For SNVs, multiple callers
27 such as GATK mutect2, VarScan2 [36], bcftools and VarDict [37] were combined to achieve a sensitivity of
28 97.5%, precision of 98.8% and F1-score of 98.1% for variants found in 2 or more callers. Similarly, SVs
29 were identified using multiple callers: DELLY [38], LUMPY [39], WHAM [40], CNVkit [41] and Svaba [42]
30 and combined using MetaSV [43] to achieve a sensitivity of 84.6%, precision of 92.4% and F1-score of
31 88.3% for duplication variants found in the DREAM3 dataset (For more details, refer to Supplementary
32 Material). The pipeline also integrates specific cancer tools to estimate tumour purity, tumour ploidy of
33 sample pair normal-tumour. Additional annotations are incorporated to the SNV calls using dbNSFP [18]
34 and/or Gemini [19] and quality control metrics were collected at various stages and visualized using MulitQC
35 [20]. This pipeline has 3 protocols (sv, ensemble or fastpass). Schemas of pipeline steps for the three
36 protocols can be found in Figures S7, 8 and 9.
37
38
39
40
41
42
43
44
45
46
47
48
49
50

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

52
53 The Methyl-Seq workflow is adapted from the Bismark pipeline [44]. It aligns paired-end reads with
54 botiwe2 default mode. Duplicates are removed with Picard and methylation calls are extracted using
55 bismark [44]. Wiggle tracks for both read coverage and methylation profile are generated for visualization.
56 Variants calls can be extracted from the WGBS data directly using bisSNP [45]. Bisulfite conversion rates
57 are estimated with lambda genome or from human non-CpG methylation directly. Several metrics based
58
59
60
61
62
63
64
65

1
2
3
4 on IHEC requirements are also calculated. Methyl-Seq can also process capture data if provided with a
5 capture bed file. [A schema of pipeline steps can be found in Figure S10.](#)

8 Hi-C Pipeline:

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

20 The Metagenomic Pipeline (rRNA gene amplification analysis):

22 This pipeline is based on the established Qiime procedure [50] for amplicon-based metagenomics.
23 It assembles read pairs using FLASH [51], detects chimeras with uchime [52] and picks OTUs using vsearch
24 [53]. OTUs are then aligned using PyNAST [54] and clustered with FastTree [55]. Standard diversity indices,
25 taxonomical assignments and ordinations are then calculated and reported graphically. [A schema of
26 pipeline steps can be found in Figure S13.](#)

31 The PacBio Pipeline:

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

52 **Comparison with other solutions for NGS analysis**

54 Data collected for select tools, modified from Griffith & Griffith et al. [57] (Table 1), shows that
55 GenPipes' strength lies in its robust WMS that comes with one of the most diverse selection of analysis
56 pipelines which have been thoroughly tested. The pipelines in the framework cover a wide range of
57 sequencing applications (Figure 2a). The pipelines are end-to-end workflows running complete
58
59
60
61

1
2
3
4 bioinformatics analyses. While many available pipelines conclude with a bam file or run limited post-bam
5 analysis steps, the pipelines included in GenPipes are extensive, often having as many as 40 different
6 steps that cover a wide range of post-bam processing. **It is important to note that GenPipes, as well as**
7 **several other WMSs, have community-supported pipelines, however, those have not been included in the**
8 **comparison.**

9
10
11 GenPipes is compatible with HPC computing, as well as cloud computing [58] and includes a
12 workflow manager that can be adapted to new systems. GenPipes also provides job status tracking through
13 JSON files that can then be displayed on a web portal (an official portal for GenPipes will be released soon).
14 GenPipes' available pipelines facilitate bioinformatics processing, while the framework makes it flexible for
15 modifications and new implementations.

16
17 GenPipes developers offer continuous support through a Google forum page [59] and a help desk
18 email address (pipelines@computationalgenomics.ca). Since the release of version 2.0.0 in 2014, a
19 community of users has run GenPipes to conduct approximately 3000 analyses processing around 100,000
20 samples (Figure 2b-c).

21 22 23 24 25 26 27 28 **DISCUSSION and CONCLUSION**

29
30 GenPipes is a workflow management system that facilitates building robust genomic workflows.
31 GenPipes is a unique solution which combines both a framework for development and end-to-end analysis
32 pipelines for a very large set of genomics fields. The efficient framework for pipeline development has
33 resulted in a broad community of developers with over 30 active branches and more than 10 forks of the
34 GenPipes repository. GenPipes has several optimized features that adapt it to large scale data analysis,
35 namely:

- 36
37
38
39
40 ● **Multiple schedulers:** GenPipes is optimized for HPC processing. It currently accommodates 4
41 schedulers.
- 42
43 ● **Job dependencies:** GenPipes establishes dependencies among its different steps. This enables
44 launching all the steps at the same time and minimizes queue waiting time **and management.**
- 45
46 ● **Smart relaunch:** GenPipes sets and detects flags at each successful step in the pipeline. This
47 allows the detection of successfully completed steps and easy relaunch of failed steps.
- 48
49 ● **Parameter encapsulation:** Genpipes uses a superposed configuration system to parse all
50 required parameters from configuration files. This simplifies the use of the framework and makes
51 it more flexible to user adjustments. Tested configuration files that are tailored to different clusters
52 and different species are included with GenPipes.
- 53
54 ● **Diverse inputs:** GenPipes has been developed to launch using different starting inputs, making
55 it more flexible.
- 56
57 ● **Flexible workflows:** GenPipes implements a workflow in steps. Users can choose to run specific
58 steps of interest, limiting waste of time and resources.

1
2
3
4
5
6 GenPipes is under continuous development to update established pipelines and to create new
7 pipelines for emerging technologies. For instance, new genomics pipelines are being developed for ATAC-
8 Seq, single cell RNA-Seq and HiChIP. GenPipes is also being redeveloped to use the **Common Workflow**
9 **Language (CWL)** to provide a cloud compatible version more seamlessly **and more Scheduler objects, like**
10 **DRMAA, are being added to expand compatibility with more platforms.** GenPipes has become a reliable
11 bioinformatics solution that has been used in various genomics publications for DNA-Seq [60-67], RNA-
12 Seq [68] and ChIP-Seq [69] analyses. GenPipes is currently available as source code, as well as a Docker
13 image for easy installation and use. GenPipes has been optimized for HPC systems but can run on a laptop
14 computer on small datasets.
15
16
17
18
19
20
21

22 **Availability and requirements**

- 23 ● Project name: GenPipes
 - 24 ● Project home page: <http://www.c3g.ca/genpipes>
 - 25 ● Operating system(s): Linux; Can be used on Windows and Mac OS using Docker
 - 26 ● Programming language: Python
 - 27 ● Other requirements: Workflow-dependant; detailed in documentation
 - 28 ● License: GNU GPLv3
 - 29 ● **SciCrunch RRID: SCR_016376**
- 30
31
32
33
34
35
36
37

38 **SUPPLEMENTARY DATA**

39 No Supplementary Data
40
41
42
43
44

45 **ACKNOWLEDGEMENT**

46
47 Data analyses were enabled by compute and storage resources provided by Compute Canada and
48 Calcul Québec. Authors would also like to acknowledge Romain Gregoire and Tushar Dubey for their
49 contribution to the code **and Patricia Goerner-Potvin for her help in planning the report content.**
50
51
52
53
54
55
56
57
58

59 **FUNDING**

60
61
62
63
64
65

1
2
3
4 This work was supported by CANARIE, Compute Canada and Genome Canada. Additional support came
5 from a grant from the National Sciences and Engineering Research Council (NSERC-448167-2013) and
6 a grant from the Canadian Institute for Health Research (CIHR-MOP-115090). GB is also supported by
7 the Fonds de Recherche Santé Québec (FRSQ-25348).
8
9

10 11 12 13 **CONFLICT OF INTEREST**

14
15
16 The Authors declare no conflict of interest.
17
18
19

20 21 **REFERENCES**

- 22
- 23 1. ENCODE, *The ENCODE (ENCyclopedia Of DNA Elements) Project*. Science, 2004. **306**(5696): p. 636-40.
- 24 2. Stunnenberg, H.G. and M. Hirst, *The International Human Epigenome Consortium: A Blueprint for Scientific*
25 *Collaboration and Discovery*. Cell, 2016. **167**(5): p. 1145-1149.
- 26 3. Mardis, E.R., *The \$1,000 genome, the \$100,000 analysis?* Genome Med, 2010. **2**(11): p. 84.
- 27 4. Afgan, E., et al., *The Galaxy platform for accessible, reproducible and collaborative biomedical analyses:*
28 *2016 update*. Nucleic Acids Res, 2016. **44**(W1): p. W3-W10.
- 29 5. DNANexus_website, <https://www.dnanexus.com/>.
- 30 6. Koster, J. and S. Rahmann, *Snakemake--a scalable bioinformatics workflow engine*. Bioinformatics, 2012.
31 **28**(19): p. 2520-2.
- 32 7. Di Tommaso, P., et al., *Nextflow enables reproducible computational workflows*. Nat Biotechnol, 2017.
33 **35**(4): p. 316-319.
- 34 8. Sadedin, S.P., B. Pope, and A. Oshlack, *Bpipe: a tool for running and managing bioinformatics pipelines*.
35 Bioinformatics, 2012. **28**(11): p. 1525-6.
- 36 9. Cingolani, P., R. Sladek, and M. Blanchette, *BigDataScript: a scripting language for data pipelines*.
37 Bioinformatics, 2015. **31**(1): p. 10-6.
- 38 10. Compute_Canda, [https://www.compute canada.ca/research-portal/account-management/apply-for-an-](https://www.compute canada.ca/research-portal/account-management/apply-for-an-account/)
39 [account/](https://www.compute canada.ca/research-portal/account-management/apply-for-an-account/).
- 40 11. P. Buncic, C.A.S., J. Blomer, L. Franco, A. Harutyunian, P. Mato, and Y. Yao., *CernVM - a virtual software*
41 *appliance for LHC applications*, in *Journal of Physics*. 2010. p. 042003.
- 42 12. GenPipes_tutorial, <http://www.computationalgenomics.ca/tutorials/>.
- 43 13. Breeze, C.E., et al., *eFORGE: A Tool for Identifying Cell Type-Specific Signal in Epigenomic Data*. Cell
44 Rep, 2016. **17**(8): p. 2137-2150.
- 45 14. Van der Auwera, G.A., et al., *From FastQ data to high confidence variant calls: the Genome Analysis Toolkit*
46 *best practices pipeline*. Curr Protoc Bioinformatics, 2013. **43**: p. 11 10 1-33.
- 47 15. Li, H. and R. Durbin, *Fast and accurate short read alignment with Burrows-Wheeler transform*.
48 Bioinformatics, 2009. **25**(14): p. 1754-60.
- 49 16. Li, H., et al., *The Sequence Alignment/Map format and SAMtools*. Bioinformatics, 2009. **25**(16): p. 2078-9.
- 50 17. Zook, J.M., et al., *Extensive sequencing of seven human genomes to characterize benchmark reference*
51 *materials*. Sci Data, 2016. **3**: p. 160025.
- 52 18. Liu, X., et al., *dbNSFP v3.0: A One-Stop Database of Functional Predictions and Annotations for Human*
53 *Nonsynonymous and Splice-Site SNVs*. Hum Mutat, 2016. **37**(3): p. 235-41.
- 54 19. Paila, U., et al., *GEMINI: integrative exploration of genetic variation and genome annotations*. PLoS Comput
55 Biol, 2013. **9**(7): p. e1003153.
- 56 20. Ewels, P., et al., *MultiQC: summarize analysis results for multiple tools and samples in a single report*.
57 Bioinformatics, 2016. **32**(19): p. 3047-8.
- 58 21. Dobin, A., et al., *STAR: ultrafast universal RNA-seq aligner*. Bioinformatics, 2013. **29**(1): p. 15-21.
59
60
61
62
63
64
65

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.

- 1
2
3
4 52.Edgar, R.C., et al., *UCHIME improves sensitivity and speed of chimera detection*. Bioinformatics, 2011. **27**(16): p. 2194-200.
- 5
6 53.Rognes, T., et al., *VSEARCH: a versatile open source tool for metagenomics*. PeerJ, 2016. **4**: p. e2584.
- 7 54.Caporaso, J.G., et al., *PyNAST: a flexible tool for aligning sequences to a template alignment*.
8 Bioinformatics, 2010. **26**(2): p. 266-7.
- 9 55.Price, M.N., P.S. Dehal, and A.P. Arkin, *FastTree: computing large minimum evolution trees with profiles
10 instead of a distance matrix*. Mol Biol Evol, 2009. **26**(7): p. 1641-50.
- 11 56.Hunt, M., et al., *Circlator: automated circularization of genome assemblies using long sequencing reads*.
12 Genome Biol, 2015. **16**: p. 294.
- 13 57.Griffith, M., et al., *Genome Modeling System: A Knowledge Management Platform for Genomics*. PLoS
14 Comput Biol, 2015. **11**(7): p. e1004274.
- 15 58.GenPipes_Cloud, <http://www.computationalgenomics.ca/genpipes-in-the-cloud/>.
- 16 59.GenPipes_GoogleForum, <https://groups.google.com/forum#!forum/GenPipes>.
- 17 60.Buczkwicz, P., et al., *Genomic analysis of diffuse intrinsic pontine gliomas identifies three molecular
18 subgroups and recurrent activating ACVR1 mutations*. Nat Genet, 2014. **46**(5): p. 451-6.
- 19 61.Scelo, G., et al., *Variation in genomic landscape of clear cell renal cell carcinoma across Europe*. Nat
20 Commun, 2014. **5**: p. 5135.
- 21 62.Le Guennec, K., et al., *17q21.31 duplication causes prominent tau-related dementia with increased MAPT
22 expression*. Mol Psychiatry, 2017. **22**(8): p. 1119-1125.
- 23 63.Torchia, J., et al., *Integrated (epi)-Genomic Analyses Identify Subgroup-Specific Therapeutic Targets in
24 CNS Rhabdoid Tumors*. Cancer Cell, 2016. **30**(6): p. 891-908.
- 25 64.Oliazadeh, N., et al., *Identification of Elongated Primary Cilia with Impaired Mechanotransduction in
26 Idiopathic Scoliosis Patients*. Sci Rep, 2017. **7**: p. 44260.
- 27 65.Bellenguez, C., et al., *Contribution to Alzheimer's disease risk of rare variants in TREM2, SORL1, and
28 ABCA7 in 1779 cases and 1273 controls*. Neurobiol Aging, 2017. **59**: p. 220.e1-220.e9.
- 29 66.Hamdan, F.F., et al., *High Rate of Recurrent De Novo Mutations in Developmental and Epileptic
30 Encephalopathies*. Am J Hum Genet, 2017. **101**(5): p. 664-685.
- 31 67.Monlong, J., et al., *Global characterization of copy number variants in epilepsy patients from whole genome
32 sequencing*. PLoS Genet, 2018. **14**(4): p. e1007285.
- 33 68.Manku, G., et al., *Changes in the expression profiles of claudins during gonocyte differentiation and in
34 seminomas*. Andrology, 2016. **4**(1): p. 95-110.
- 35 69.Deblois, G., et al., *ERRalpha mediates metabolic adaptations driving lapatinib resistance in breast cancer*.
36 Nat Commun, 2016. **7**: p. 12156.
- 37 70.Fisch, K.M., et al., *Omics Pipe: a community-based framework for reproducible multi-omics data analysis*.
38 Bioinformatics, 2015. **31**(11): p. 1724-8.
- 39 71.Reich, M., et al., *GenePattern 2.0*. Nat Genet, 2006. **38**(5): p. 500-1.
- 40 72.O'Connor, B.D., B. Merriman, and S.F. Nelson, *SeqWare Query Engine: storing and searching sequence
41 data in the cloud*. BMC Bioinformatics, 2010. **11 Suppl 12**: p. S2.
- 42 73.Buske, F.A., et al., *NGSANE: a lightweight production informatics framework for high-throughput data
43 analysis*. Bioinformatics, 2014. **30**(10): p. 1471-2.
- 44 74.Ceraj, I., Riley, J. T., Shubert, C., *StarHPC - Teaching Parallel Programming within Elastic Compute Cloud*.
45 Proceedings of the ITI 2009 31st Int. Conf. on Information Technology Interfaces, June 22-25, 2009.
- 46 75.Taghiyar, M.J., et al., *Kronos: a workflow assembler for genome analytics and informatics*. Gigascience,
47 2017. **6**(7): p. 1-10.
- 48
49
50
51
52
53
54
55
56
57
58

59 **TABLE AND FIGURES LEGENDS**

60
61
62
63
64
65

Figure 1 - General workflow of GenPipes

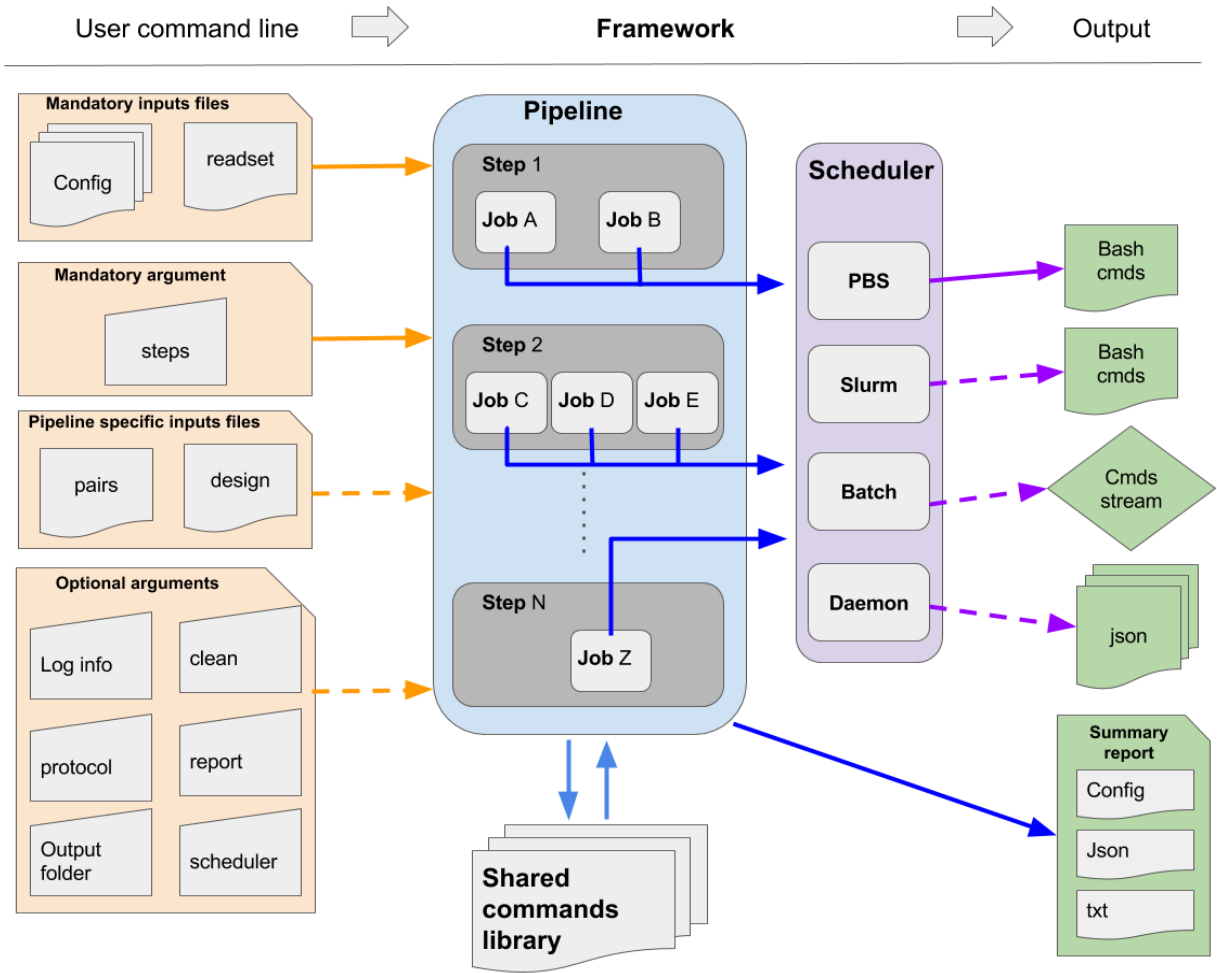
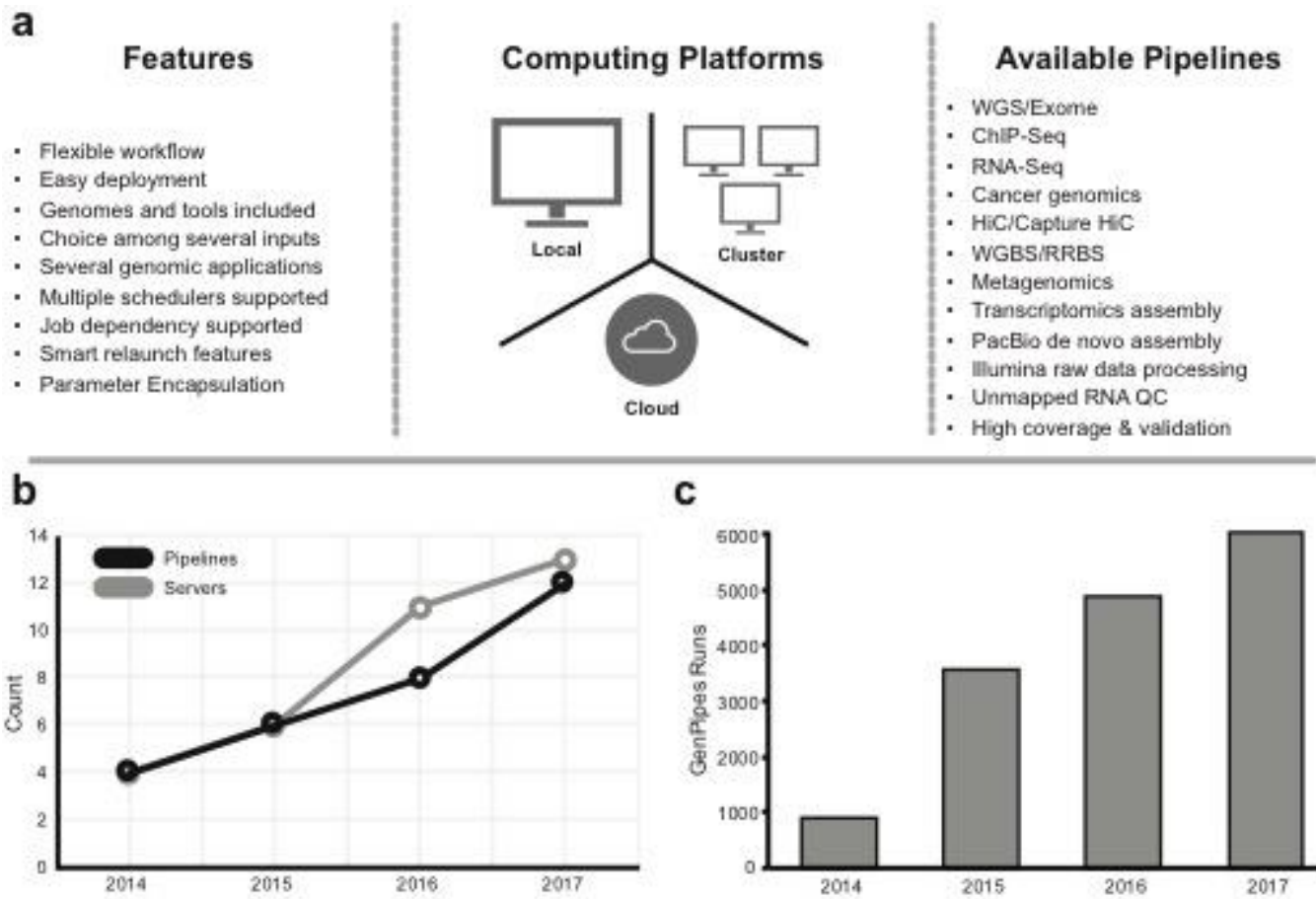


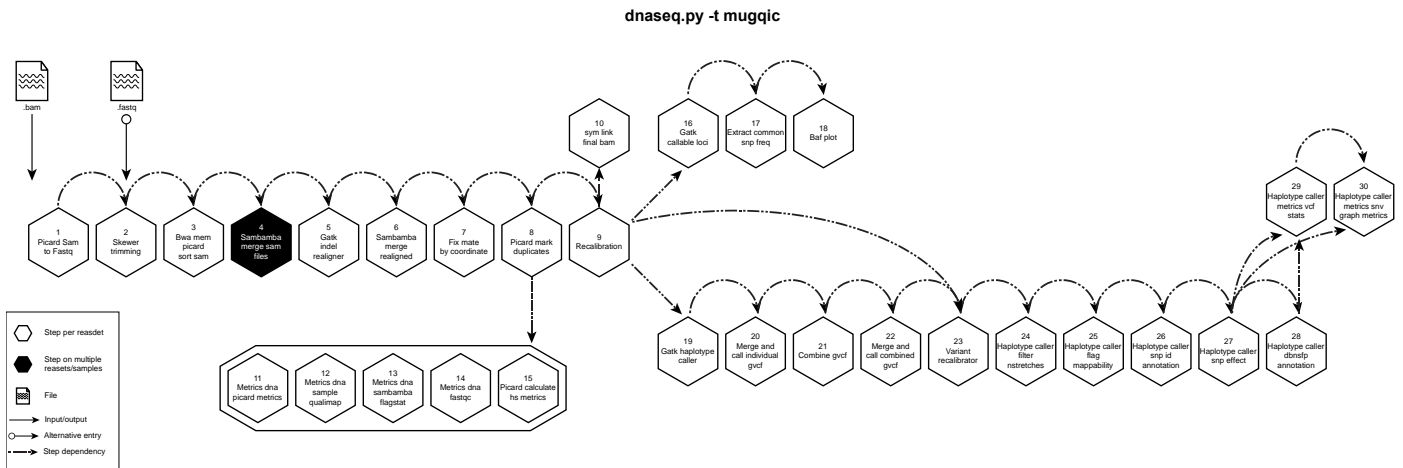
Diagram showing how the information flows from the user command line input through the 4 different objects (*Pipeline, Step, Job and Scheduler*) in order to generate system specific executable outputs.

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 DNaseSeq pipeline diagram

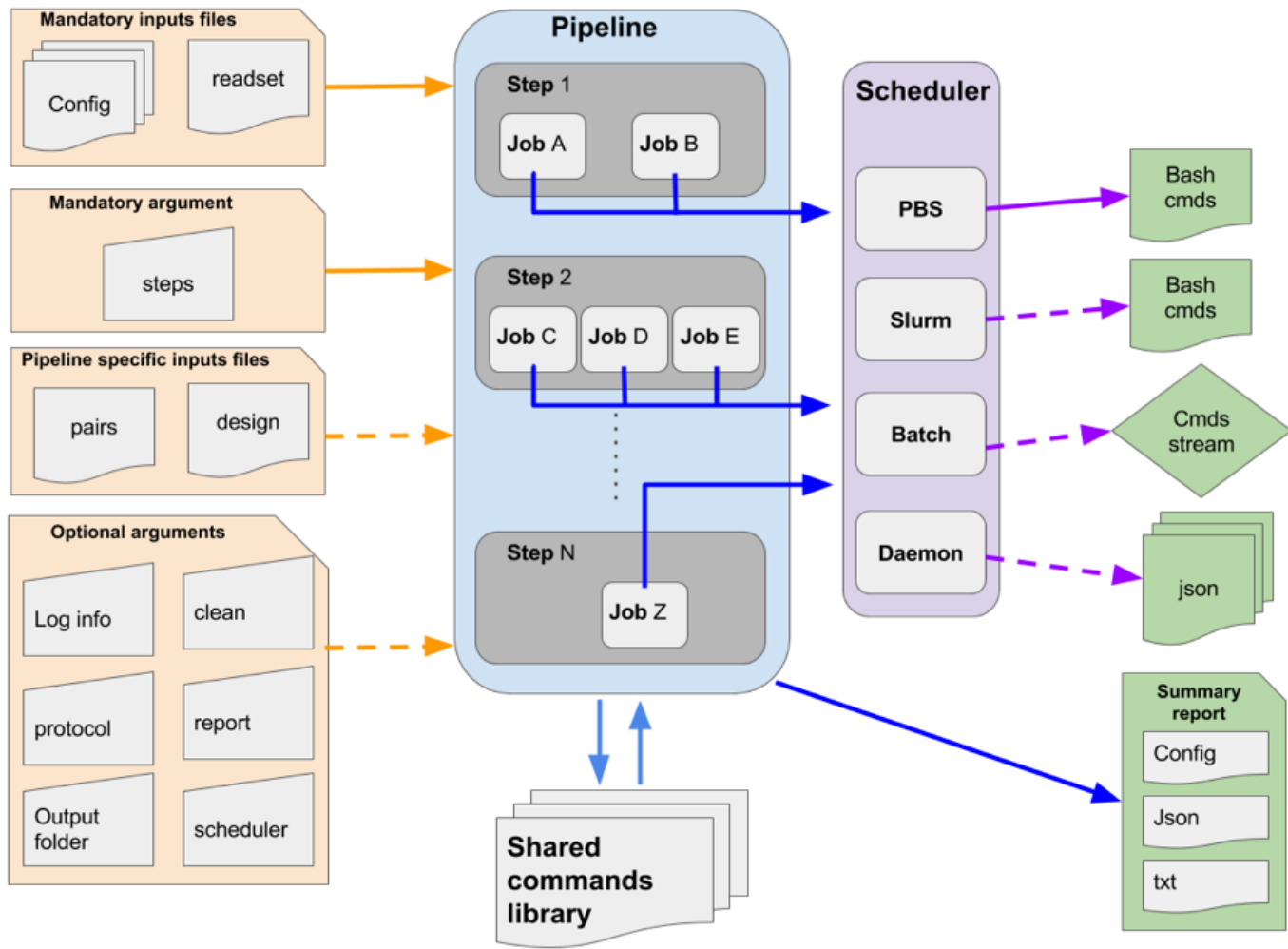


Schematic representation of GenPipes' dnaseq.py pipeline. Hexagons represent steps in the pipeline. White hexagons represent steps that process input from a single sample, while black ones represent steps that process input from several samples. Arrows show step dependencies.

Table 1 - Comparison of available solutions for NGS analysis.

Solution	Features											Pipelines										
	Language	Software license	Published	Free	Open source	Cloud/Container	HPC	Workflow manager	Tracking	GUI	Reports	Config Validation	Germline	Somatic	RNA-Seq	RNA-Seq De novo	ChIP-seq	Metagenome	Methyl-Seq	Hi-C	PacBio assembly	
GenEPI	Python	GNU LGPL	Pending	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✓
Genome Modeling System	Perl	GNU LGPLv3	[57]	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗
Galaxy	Python	Academic Free L3.0	[4]	✓	✓	✓	✓	✓	✓	✗	✗	N/A	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
bbioinformatics	Python	MIT License	No	✓	✓	✓	✓	✓	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
Omics Pipe	Python	MIT License	[70]	✓	✓	✓	✓	✓	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
Gene Pattern	Java	Custom	[71]	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	N/A	✓	✗	✗	✗	✗	✗	✗	✗
Illumina BaseSpace	bash	Custom	No	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗
BINA Genomic Analysis	Java/Python	Custom	No	✗	✗	✓	✓	✓	✓	N/A	N/A	N/A	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗
SeqWare	Java	GNU GPLv3	[72]	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
DNA Nexus Platform	Python/bash	Custom	No	✓	Partial	✓	✗	✓	✓	✓	✓	N/A	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
gkns	Python	MIT License	No	✓	✓	✓	✓	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
NGS2H	bash	BSD3	[73]	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
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 S2AD	Python	GNU GPLv3	[74]	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
CronW/WDL	Scala	BSD 3-Clause	No	Partial	✓	✓	✓	✓	✓	✗	✗	N/A	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗
BioDataScript	BDS	Apache License V2	[9]	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
Kronos	Python	MIT license	[75]	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
Nextflow	Java	GNU GPLv3	[7]	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗
Snakemake	Python	MIT License	[6]	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗

Modified from Griffith & Griffith et al. [57].

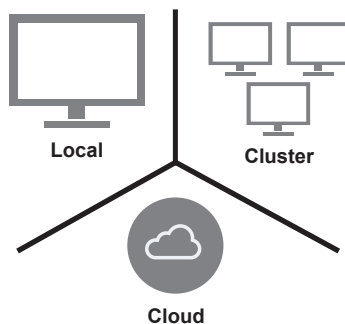


a

Features

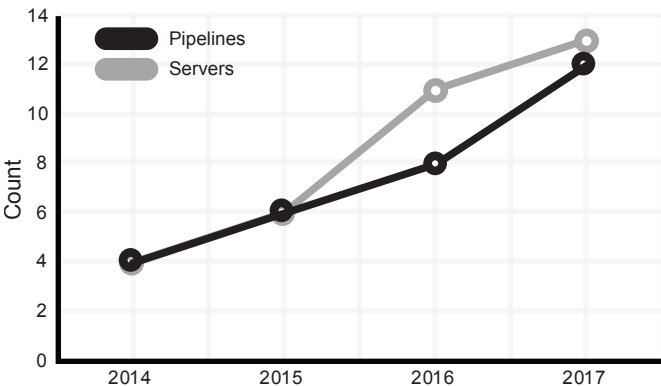
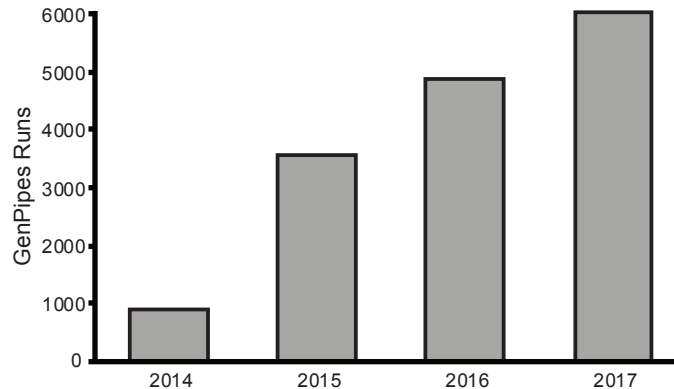
- Flexible workflow
- Easy deployment
- Genomes and tools included
- Choice among several inputs
- Several genomic applications
- Multiple schedulers supported
- Job dependency supported
- Smart relaunch features
- Parameter Encapsulation

Computing Platforms

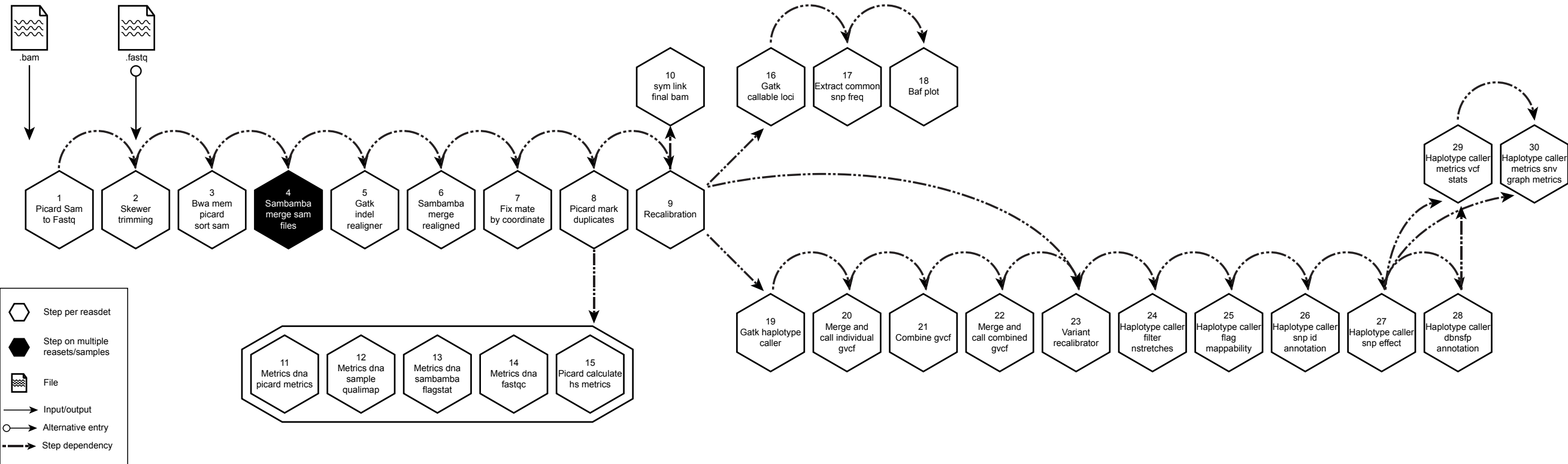


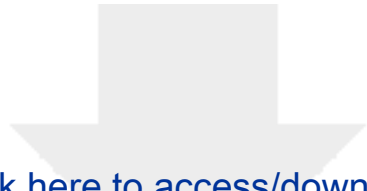
Available Pipelines

- 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 muggic



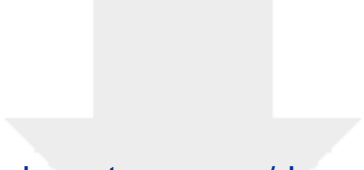


Click here to access/download
Supplementary Material
Supplementary_Materials.docx

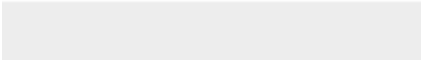





Click here to access/download
Supplementary Material
FigureS1.pdf



Click here to access/download
Supplementary Material
TableS1.xlsx



Dear GigaScience Editorial Office,

Thank you for opportunity to submit a revised version of the manuscript GIGA-D-18-00198 that addresses the points raised by the two reviewers. Following the constructive comments and suggestions, we have made several major improvements to the manuscript, including:

- The reviewers had comments about the dependencies between steps, as well as whether steps worked on single samples or a cohort of samples. To help answer these questions, we have added a diagram summarizing each pipeline to the supplementary.
- We have added extensive benchmark metrics for two pipelines, dnaseq and tumour_pair.
- We have added documentation on how to run *GenPipes* on a cloud platform on our website and now referenced it in the manuscript.
- *GenPipes* is now registered in the SciCrunch database under the RRID *SCR_016376*. The RRID has been included in the revised manuscript under the ‘*Availability and requirements*’ section.
- We have re-organized the text a bit to better explain the unique features of *GenPipes*

See below for our point-by-point response to the reviewers. New text that has been added is shown in red in the revised manuscript.

Response to the Reviewers:

Reviewer #1: The authors present in this manuscript both a new workflow management system (GenPipe), as well as a set of bioinformatics pipelines that are built to run on this system. The authors contribution is likely be of interest to many genome centres and bioinformaticians, who wish to leverage existing pre-built and tested pipelines. The manuscript is clear and well written and the source code is well structured, extensive, and is well documented. The developers have also taken steps to ease installation and configuration issues that might occur when trying to install the software in other environments.

However, I have reservations regarding the structure and content of the manuscript. I find it lacks detail and analysis that would convince a reader to adopt their system, both with regards to GenPipe itself, as well as the pipelines. This is unfortunate as I think the authors have provided a large contribution to the field in making available their resources.

We thank the reviewer for evaluating the manuscript and for his positive view of our contribution to the bioinformatics community. We have now re-organized the text a bit to better explain the unique features of *GenPipes* and have added more information to the manuscript that should provide the users with a better overview of its advantages (see below).

Major points:

1. The authors only provide a superficial comparison to existing systems. A more detailed analysis of why new pipeline developers should use GenPipe over an alternatives? What distinguishes this as a WMS from SnakeMake for example? From what I can see in the manuscript there are several implementation details within GenPipe that appear sub-optimal, which I'll elaborate on in the points below.

In our original submission, we attempted to provide a comparison of *GenPipes* to other available WMSs. While far from exhaustive, we had compared *GenPipes* to 18 popular WMSs by looking at 9 different features and 9 different pipelines (Table 1). In the revised submission, we have added SnakeMake to the WMS list and included 3 more features suggested by reviewer #2 (see below). We have also added a detailed workflow for each pipeline (Figure 3 and S1-14) and have added new benchmarking metrics for some of the pipelines. We also included estimate resource usage for each pipeline across different servers (Table S1). We have also moved the description of the unique features of *GenPipes* to the Result section of the manuscript. We think that this new information will hopefully give potential users a better idea of how *GenPipes* compares to some of the existing frameworks available and its advantages.

2. I would also like to see a proper analysis for each pipeline (can be provided in supplemental information) describing comparisons to existing pipelines in terms of accuracy, resource usage, runtime stats, etc.

We agree with the reviewer that benchmarking the individual pipelines for accuracy, resource usage and runtime statistics is important. Along those lines, we have added benchmarks for several pipelines to the Supplementary Material. In terms of resource usage and runtime statistics, it has been our experience that these metrics vary widely depending on system hardware, software versions and sample sequencing depth. However, we agree that a ballpark estimate of these resources would be useful for the user and have added Table S1.

In terms of accuracy, it is important to remember that *GenPipes* is a framework that is built around open source, third party tools that are available to the scientific community. Accuracy is not as easily assessed in certain fields due to the lack of a good quality “truth” set to benchmark against. Generally, we have tried to model *GenPipes* pipelines following large scale projects like GATK best practices SOPs and ENCODE and have relied on public benchmarking, in addition to our own.

Minor points:

1. Introduction: "Such solutions are flexible and can help in pipeline implementation but do not provide robust standardized pipelines which are ready for production-scale analysis." In my experience, it's simply not true that WMS solutions are not suitable for production scale analysis.

There are many examples of people doing exactly this, and moreover I've built several myself which are run multiple times every day without issue. In my experience they can work very reliably, and ability to tolerate and resume from errors is easy to code in. It is also unclear what the authors mean by standardisation in this context. I'd request that the authors either justify this point and provide concrete examples of exactly what the source of the perceived issues are, or remove this sentence.

We agree with the reviewer that the sentence is not delivering the idea we intended. The sentence has been edited in the manuscript to highlight the fact that not all WMSs come with **pre-built** pipelines that are ready for use. We agree with the reviewer that many WMSs are robust and powerful. We are simply trying to appeal to both the advanced user with *GenPipes*'s WMS and the novice user with the pre-built and tested pipelines.

The sentence now reads:

“Such solutions are flexible and can help in pipeline implementation but *rarely* provide robust *pre-built* pipelines which are ready for production analysis.”

2. Introduction: "These are useful for specific applications but can be challenging to implement, difficult to modify or scale-up. They have also rarely been tested on multiple computing infrastructures." This seems too strong a statement. In some cases this might be true but there are many examples of robust pipelines that efficiently leverage data centre hardware.

We agree with the reviewer that the statement might be generalized beyond context. We have edited the text to explicitly prevent the statement from applying to all pipelines.

The sentence now reads:

“These are useful for specific applications but can *sometimes* be challenging to implement, difficult to modify or scale-up.”

3. Introduction: "GenPipes has been tested, benchmarked ...". It is not clear whether the "testing and benchmarking" refers to the pipelines or the WMS itself. This should be clarified.

Both *GenPipes*'s pipelines and the WMS have been tested extensively. The pipelines have been benchmarked and have been used to process thousands of samples. The WMS has been stress-tested and adapted to different computing infrastructure and is currently run on at least 6 super computers that we help maintain. We have modified the text to clarify this.

The sentence now reads:

“*GenPipes*' WMS and pipelines have been tested, benchmarked and used extensively over the past four years.”

4. Schedulers: Ideally, GenPipes should offer the ability to implement scheduling via DRMMMA which would increase the potential sites that could potentially run genpipe. For example, currently

any data centres running Platform LSF could not use genpipe but via4 DRMAA this would be possible.

We thank the reviewer for his suggestion. We generally like to minimize the number of layers between *GenPipes* and the system it is running on, and did not consider DRMAA previously. We have added the DRMAA scheduler to our potential future projects and have mentioned it in the discussion.

5. Job dependencies: I have reservations that the approach taken here is optimal. If I understand correctly, job dependencies are setup using the selected scheduler and all jobs, across steps, are launched at the same time. I suspect for very large pipelines containing many thousands jobs (not uncommon) this would put an undue burden on the scheduler and therefore would not scale very well. Could the authors elaborate on this point and highlight details such as what happens when a pipeline fails? Are existing jobs explicitly terminated? Or somehow left running and continue after the pipeline is resumed?

We agree with the reviewer that this feature may be optimized and have been working on this for a future *GenPipes* release. It is not ready yet, as we would like to optimize and test all supported schedulers before releasing the new feature. The current process works by creating the full script which is communicated to the scheduler. This approach has been initially chosen for its low level of complexity and reproducibility; the script is readable and editable by a minimally trained user and can also be re-run later on. This also avoid having to generate local processes which would need to stay in active mode in order to monitor the job submission process. In order to help monitoring the pipeline progress, *GenPipes* includes a script (`logReport.pl`) that generates a report of the current status of the pipeline. We have also included a JSON log file system that could be used to develop a local web-portal displaying the pipeline job status in real time. We have been using *GenPipes* on hundreds of samples every day, submitting thousands of jobs, and has not run into any issues with our compute providers so far. However, we do intend to optimize the process by using job arrays.

In terms of what happens when the pipeline fails, we have added a section to the text under “Running GenPipes” to elaborate on this point:

“... 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...”

6. Configuration Files: "Configuration files, also referred to as "ini" files, are provided among the arguments of the GenPipes command.". The authors should change the wording here. "Ini" is a legacy windows-based configuration file format. I'm not asking for the authors to change the configuration format used but it would be useful to have some justification for this unusual

choice. Alternatives, like "yaml" for example allow for stricter and richer structuring and is therefore much easier to parse and in turn normally results in less buggy code.

We agree with the reviewer that many file configuration formats exist and that some might be considered more optimal than the legacy ini file format. However, we chose the ini format for its readability and the ease of use of the ini schema (section, key and value). It can be edited manually with any text editor without the need to worry about syntax or indentation which is easier for standard analysts using *GenPipes* for their analysis. Additionally, the python language offers a standard library (configParser) that is made to easily integrate this standard configuration format.

Reviewer #2: The manuscript presents GenPipes, a Python-based framework for defining and executing data analysis workflows.

GenPipes is based on a handful of Python classes that can be inherited and implemented to achieve a formal and executable description of a workflow in terms of steps. During execution, steps are specialized to jobs that perform concrete operations on input files.

For me, the most important, and definitely valuable addition of this work is the comprehensive collection of well-tested workflows covering the most important applications of sequencing.

In general, I think this should be emphasized more, at the expense of removing some of the weaker aspects of the paper. I will outline this below.

Major Comments

* The manuscript argues that a major advantage of GenPipes is the rich collection of production-ready workflows that are delivered with the system. The list of workflows is indeed impressive, it should be mentioned though that both Snakemake and Nextflow also provide (community-maintained) collections of tested workflows, like github.com/snakemake-workflows, nf-core.github.io and sequana. I agree though that it might very well be that these are still less mature (except sequana), as they are probably newer.

We thank the reviewer for seeing value in our growing collection of pipelines. In our manuscript, we avoided reference to community-maintained workflows. While community-maintained workflows are a great testament to the usefulness of a WMS, they are hard to keep track of and evaluate. *GenPipes* supports community-maintained workflows as well, however, those too have not been mentioned in the manuscript. The pipelines that have been mentioned are pipelines that have been validated and are maintained by the tool authors, which we think is an important distinction.

We have made this clear in the text, as follows:

“It is important to note that GenPipes, as well as several other WMSs, have community-supported pipelines, however, those have not been included in the comparison.”

* The manuscript claims that GenPipes supports cloud execution, but I cannot find a scheduler for this purpose in the list of schedulers on page 4. Also, the feature table says that cloud support is pending.

Yes, *GenPipes* supports cloud execution via a container image and not a particular scheduler. Through the container image, any available scheduler can be used, depending on the cloud architecture in place. We apologize for the omission in Table1, we have fixed it. We have now also documented the use of *GenPipes* in the cloud and added a user manual at: <http://www.computationalgenomics.ca/genpipes-in-the-cloud/>

* On page 7, when describing deployment of software and reference information, it is unclear whether installation happens system wide (needing admin rights) or local. This should be clearly stated, since system-wide installation would be a major disadvantage compared to systems like Nextflow, Snakemake or CWL based WMSs. Moreover, it should be mentioned how those dependencies are updated, and in what sense such updates would affect previous runs, which could potentially lose reproducibility, if updates happen globally.

GenPipes installation can happen both system-wide or locally as the pipeline will only use software and modules provided in a specific path defined through an environment variable. All the third part tool installation scripts provided with the pipeline have been designed to work on a local path system and do not require root privileges. We also developed a container image of *GenPipes* which runs *GenPipes* with little software installation. This allows larger processing centers to install *GenPipes* for all users, but also allows individuals the flexibility to adopt *GenPipes* for their own needs without needing special permissions or setup.

We have added text in the result section to explain these points and to expand on the dependency updates:

“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.”

* In the discussion, it is mentioned that GenPipes is currently being reimplemented in WDL. It is a good choice to use one of the established, more feature-rich systems. However, then, large parts of this paper are in fact obsolete, as they will be replaced with WDL. The major contribution that remains after that step is the collection of workflows, which is totally fine, since this is a very valuable addition. I therefore suggest to put more focus on the workflows, and simply outline that they are currently implemented in GenPipes and soon will be available in WDL. Moreover, choice of tools, parameters and how the benchmarking was done (in a more concrete way instead of simply saying "we used GIAB") should be described in detail.

Actually, based on recent developments, we are probably going to go with CWL over WDL. That being said, we do not fully agree with the reviewer that this will make *GenPipes* WMS obsolete. While CWL or WDL can help increase the compatibility of *GenPipes* with different systems, it will add a layer of complexity to *GenPipes*; one that is not needed on HPC systems. We agree with the reviewer that the pipelines are a major contribution of *GenPipes* and that they deserve a more detailed description. We have now added more descriptions and benchmarks as supplementary material. We have also added workflow diagrams representing pipeline workflow and dependencies in the supplementary.

* Table 1 provides a feature comparison. As with every single feature comparison I have seen so far, it is highly biased, showing only features that *GenPipes* itself provides. For example, GUI (as provided e.g. by Galaxy) and automatic reports are missing. Per-step/job software deployment and container support is missing. Config file validation is missing. Items are not sufficiently explained (e.g., what is meant with tracking, and in what sense is Nextflow not providing it). A popular system is completely missing from the table: Snakemake. Via nf-core and other projects, Nextflow and Snakemake provide several of the mentioned pipelines. Finally, I cannot actually find that table in reference [62], although the authors claim that it is a modified version of the table from that paper.

We thank the reviewer for his excellent suggestions. We have added 3 more features to the comparison table (GUI, Reports and Config validation). We have also added SnakeMake to the comparison. In evaluating the pipelines available, we have looked at the published manuscript if it exists or the code base and documentation of each tools. Community-maintained pipelines were not considered, as it becomes difficult to draw the line on what to include in the comparison and it is hard to assess the reliability of these pipelines without extensive benchmarking. Only pipelines provided by the tool authors were considered.

Finally, the comparison Table1 was modified from Griffith et al. in supplementary material (Table S6) which contains a simple version of the table. following the link in the figure leads to the full version of the table:

<https://journals.plos.org/ploscompbiol/article/file?id=info%3Adoi/10.1371/journal.pcbi.1004274.s021&type=supplementary>

Minor Comments

* On page 2, when mentioning other WMSs, the authors should also mention Nextflow. Moreover, CWL and WDL are not WMSs, and should be listed separately as "declarative workflow description languages".

We have edited the text as suggested.

* On page 5, when relaunch features are mentioned, common functions of other systems like manual forcing or handling of missing files are not mentioned. Are these not available?

GenPipes supports manual forcing through the “-f” option. GenPipes validates the existence of all required modules and genome files and input files in the config file before creating the commands. If any are missing, it looks for alternative files using the ordered list of input implementation described in the “Key GenPipes features options”. If none of the possible files are found, GenPipes will throw a Missing File exception and terminate. The text has been edited to clarify this.

* On page 6, the description of input choice does not really make it clear how multiple input files or aggregation is handled. It would be beneficial to see examples for (a) a 1-in-1-out job, (b) an aggregating job, (c) a scattering job, (d) a mixed job (n-in-m-out).

GenPipes has an array of steps with different behaviors. Some steps operate on a single sample input while others operate on the cohort of available samples (metric steps). To try to distinguish these, we added color coding (black/white) to the workflow diagrams we added in supplementary Figure 1. The dependencies between steps are mapped by *GenPipes* through input and output files required and communicated to the scheduler which then coordinated job launch. For a full list of pipelines and steps, please refer to Figure S1-14.

* On page 8: "all workflows accepts a bam or fastq file as input". I guess they accept multiple bams or fastqs, right? Otherwise they could only be applied to a single sample at a time...

To take full advantage of HPC power and reduce processing times, *GenPipes* runs each sample separately, when possible. Some steps aggregate inputs from many samples at a time. Those have been colored in black in supplementary Figure2 S1-14.